A MULTI-FIDELITY ANALYSIS SELECTION METHOD USING A CONSTRAINED DISCRETE OPTIMIZATION FORMULATION

A Thesis
Presented to
The Academic Faculty

by

Ian C. Stults

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
December 2009

Copyright © 2009 by Ian C. Stults

A MULTI-FIDELITY ANALYSIS SELECTION METHOD USING A CONSTRAINED DISCRETE OPTIMIZATION FORMULATION

Approved by:

Professor Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Assistant Professor Brian German
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Don Beeson
General Electric
Technology Infrastructure

Dr. Viren Kumar
General Electric
Energy Infrastructure

Dr. Scott Duncan
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: 14 August 2009

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dimitri Mavris. Your guidance and suggestions have been very helpful. Thank you for showing me some of the soft skills associated with being a successful engineer. I sincerely appreciate the opportunity to have been a part of the ASDL family for the last 8 years. It has been a truly memorable experience. I also want to thank my thesis committee: Brian German, Don Beeson, Scott Duncan, and Viren Kumar. I have really appreciated your probing questions, feedback, guidance, and time over the last 2 years as I have worked on this research; your input has played a large part in shaping this into what it is today.

Thank you to William Engler, Santiago Balestrini, and Hernando Jimenez at ASDL for your help along the way. Your feedback and suggestions during the Ph.D. process has been especially helpful. Thanks guys; I couldn't have done it without your help. At ASDL I would also like to thank my research team: Matt Prior and Matt Daskilewicz. Your thoughts and criticisms were helpful, especially in the early stages of forming my topic.

I want to thank my wife, Sara. Thank you for your constant support and especially patience as I've struggled along the way from finding a topic to late nights launching experiments and proof-reading. I don't know that this would have been possible without it. Thank you to Mom, Dad and Brennan for your help and support through the process.

Ian

June 2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ABC Artificial Bee Colony

ACO Ant Colony Optimization

ACO-MO Ant Colony Optimization - Multi-Objective

AFRL Air Force Research Laboratory

AI Artificial Intelligence

ANN Artificial Neural Network

ANOVA Analysis of Variance

ASDL Aerospace Systems Design Laboratory

ASME American Society of Mechanical Engineers

AVETeC Advanced Virtual Engine Test Cell, Inc.

BCGA Binary-Coded Genetic Algorithm

BIP Binary Integer Programming

CA Contributing Analysis

CACO Continuous Ant Colony Optimization

CACO-MO Continuous Ant Colony Optimization - Multi-Objective

CBF Cumulative Belief Function

CDF Cumulative Distribution Function

CPF Cumulative Plausibility Function

CPG Calorically Perfect Gas

DACE Design and Analysis of Computer Experiments

DoE Design of Experiments

DNA Deoxyribonucleic Acid

DSM Design Structure Matrix

CFD Computational Fluid Dynamics

CMGEN Compressor Map Generator

C-L Chen-Lind

E^3 Energy Efficient Engine

EDS Environmental Design Space

EHM Engine Health Monitoring

FA Firefly Algorithm

FAA Federal Aviation Administration

FEM Finite Element Methods

FFCACO Frontier Finding Continuous Ant Colony Optimization

FFPSO Frontier Finding Particle Swarm Optimization

FFRCGA Frontier Finding Real-Coded Genetic Algorithm

FoM Figure of Merit

FOSM First Order Second Moment

FPI Fast Probability Integration

FSP Fidelity Selection Problem

GA Genetic Algorithm

GE General Electric

GRASP Greedy Randomized Adaptive Search Procedure

GSE Global Sensitivity Equation

HPC High Pressure Compressor

HPT High Pressure Turbine

HS Harmony Search

INLP Integer Nonlinear Programming

IP Integer Programming

IS Importance Sampling

LHS Latin Hypercube Sampling

LP Linear Programming

LPT Low Pressure Turbine

LSM Local Sensitivity Matrix

LSV Local Sensitivity Vector

MC Monte Carlo

MDO Multidisciplinary Optimization

MLP Multilayer Perceptron

MODFAN Flow Modulating Fan

MOFO Multi-Objective Function Optimization

MOPSO Multi-Objective Particle Swarm Optimization

M&S Modeling and Simulation

MS Management Science

MSTI Medical Simulation Training Initiative

NASA National Aeronautics and Space Administration

NASP Non-additive Shortest Path

NESSUS Numerical Evaluation of Stochastic Structures Under Stress

NSGA Nondominated Sorting Genetic Algorithm

NSGA-II Nondominated Sorting Genetic Algorithm - II

NLP Nonlinear Programming

NP Nondeterministic Polynomial

NPSS Numerical Propulsion System Simulation

NRC Nuclear Regulatory Commission

OEC Overall Evaluation Criterion

OLH Optimal Latin Hypercube

OPR Overall Pressure Ratio

OR Operations Research

P Polynomial

PART Parametric Turbine

PC Power Code

PDF Probability Density Function

PEM Point Estimation Method

P&W Pratt and Whitney

PSO Particle Swarm Optimization

QAP Quadratic Assignment Problem

QMC Quasi Monte Carlo

RCSP Resource Constrained Shortest Path

RBF Radial Basis Function

RCGA Real-Coded Genetic Algorithm

R-F Rackwitz-Fiessler

RSE Response Surface Equation

SA Simulated Annealing

SDV Sensitivity Derivative Vector

SFTF Separate Flow Turbofan

SLS Sea-Level Static

SOSM Second Order Second Moment

SPP Shortest Path Problem

SQP Sequential Quadratic Programming

SVM Support Vector Machine

SVR Support Vector Regression

SwRI Southwest Research Institute

TASS TNO Automotive Safety Solutions

TCSP Time Constrained Shortest Path

TS Tabu Search

TSFC Thrust Specific Fuel Consumption

TSP Traveling Salesman Problem

UAPT University Research, Engineering, and Technology Institutes (URETI) for Aero-propulsion and Power Technology

URETI University Research, Engineering, and Technology Institutes

US United States

VBA Virtual Bee Algorithm

VCSP Vertex Constrained Shortest Path

VIPER-CAT Virtual Integrated Propulsion Environment for Revolutionary Concepts, Architectures, and Technologies

VR Variance Reduction

V&V Verification and Validation

SUMMARY

The purpose of this research is to develop a method for selecting the fidelity of contributing analyses in computer simulations. Model uncertainty is a significant component of result validity, yet it is neglected in most conceptual design studies. When it is considered, it is done so in only a limited fashion, and therefore brings the validity of selections made based on these results into question. Neglecting model uncertainty can potentially cause costly redesigns of concepts later in the design process or can even cause program cancellation. Rather than neglecting it, if one were to instead not only realize the model uncertainty in tools being used but also use this information to select the tools for a contributing analysis, studies could be conducted more efficiently and trust in results could be quantified. Methods for performing this are generally not rigorous or traceable, and in many cases the improvement and additional time spent performing enhanced calculations are washed out by less accurate calculations performed downstream. The intent of this research is to resolve this issue by providing a method which will minimize the amount of time spent conducting computer simulations while meeting accuracy and concept resolution requirements for results.

In many conceptual design programs, only limited data is available for quantifying model uncertainty. Because of this data sparsity, traditional probabilistic means for quantifying uncertainty should be reconsidered. This research proposes to instead quantify model uncertainty using an evidence theory formulation (also referred to as Dempster-Shafer theory) in lieu of the traditional probabilistic approach. Specific weaknesses in using evidence theory for quantifying model uncertainty are identified and addressed for the purposes of the Fidelity Selection Problem. A series of experiments was conducted to address these weaknesses using n-dimensional optimization test functions. These experiments found that model uncertainty present in analyses with 4 or fewer input variables could be effectively quantified using a strategic distribution creation method; if more than 4 input variables

exist, a Frontier Finding Particle Swarm Optimization should instead be used.

Once model uncertainty in contributing analysis code choices has been quantified, a selection method is required to determine which of these choices should be used in simulations. Because much of the selection done for engineering problems is driven by the physics of the problem, these are poor candidate problems for testing the true fitness of a candidate selection method. Specifically moderate and high dimensional problems' variability can often be reduced to only a few dimensions and scalability often cannot be easily addressed. For these reasons a simple academic function was created for the uncertainty quantification, and a canonical form of the Fidelity Selection Problem (FSP) was created. Fifteen best- and worst-case scenarios were identified in an effort to challenge the candidate selection methods both with respect to the characteristics of the tradeoff between time cost and model uncertainty and with respect to the stringency of the constraints and problem dimensionality. The results from this experiment show that a Genetic Algorithm (GA) was able to consistently find the correct answer, but under certain circumstances, a discrete form of Particle Swarm Optimization (PSO) was able to find the correct answer more quickly. To better illustrate how the uncertainty quantification and discrete optimization might be conducted for a "real world" problem, an illustrative example was conducted using gas turbine engines.

CHAPTER I

MOTIVATION

Looking at the world around us, we can see our daily life is enabled by many complex physical systems. These systems range from the power plants designed to create electricity for our homes to the cars we drive to work to the communication network we use to be a working part of society. Over time, these systems that define our lives have increased in complexity. Looking at gas turbine engines for example, we can see a significant increase in complexity from Whittle and von Ohain's first thrust producing gas turbines to those used today on civil aircraft.

As the complexity of these systems has increased through time, the cost of testing them has also significantly increased. There has also been an exponential increase in computing power in recent decades. These two facts are the problem and solution to our testing needs. Computer simulations have grown in importance and frequency in science and engineering in recent years. According to the National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science, "*there is an overwhelming concurrence that simulation is key to achieving progress in engineering and science in the foreseeable future*".[162] This statement can be supported by looking at current government and industry design work. In the medical industry, a new field, simulation-based medicine, is developing. In parts of traditional procedural medicine, training has put the patient at some risk; these are areas where computer-based simulation can have a significant impact.[191] With the average cost of surgery over $1500 / hour,[47] there is an imperative to minimize training on actual patients and conduct simulation-based training when possible. However, developments are not limited to industry; the United States (US) Army began a visionary program called the Medical Simulation Training Initiative (MSTI) to better train over 100,000 medics on wartime-specific injuries using computer simulation.[15]

Looking beyond the medical field to the automotive industry, we have also seen a trend in

recent years for car manufacturers to use computer simulations to measure crash-worthiness in automobile accidents. Companies like Denton ATD, Inc. have been manufacturing advanced crash-test dummies for years. These dummies were initially developed to save lives, where they have been quite successful. In recent years, a focus on not just saving lives but also reducing injury in crashes has become extremely important. While Denton ATD is developing advanced programs like RibEye, an advanced rib deflection dummy, they have also partnered with TNO Automotive Safety Solutions (TASS).[27] TASS is a company that develops advanced, virtual dummies for a variety of applications with the goal of making *"transport including cars, motorcycles, buses, commercial vehicles, mass transit systems and aircraft more than just safe."*[206] Through this partnership, TASS and Denton ATD, Inc. will be using virtual dummies to assess crash-worthiness of new automobiles and improving the overall safety of transportation through their use of simulation.[27]

National Aeronautics and Space Administration (NASA) and gas turbine manufacturers are working together with the Advanced Virtual Engine Test Cell, Inc. (AVETeC) to develop an advanced virtual test cell. This not-for-profit group is working in the Cleveland, OH area with General Electric (GE), Pratt & Whitney, NASA-Glenn, and the Air Force Research Laboratory (AFRL), among other groups, to develop their proof-of-concept. This development is being done in partnership with the Department of Energy using a series of supercomputers. These computers are leveraging NASA's Numerical Propulsion System Simulation (NPSS), a framework for propulsion simulation developed with a virtual engine test bed in mind, where its NPSS creators hoped to half the development time for an engine concept through detailed, integrated simulation.[201] Engine development requires a significant amount of time and money. Additionally, the testing alone may require up to 50% of the program budget, according to the AVETeC Chief Technology Officer.[189] By having a virtual test cell capability, AVETeC hopes to allow industry to significantly reduce this cost and make the development process much more efficient. By improving the efficiency of the development process, both military and civil engine applications can be developed much more easily. The benefit of a virtual engine test cell does not end with engine development and certification. Airlines operating around the world spend nearly $40 billion, 20% of their

total operating budgets, on maintenance according the the American Society of Mechanical Engineers (ASME). A good portion of this is spent maintaining their engines, and with a simulation capability like a virtual engine test cell, Engine Health Monitoring (EHM) programs would have a significantly reduced cost because of the ability for a better root cause analysis. While the case for EHM is made for the commercial side, this would be just as significant if not more so on the military side where maintenance occurs much, much more regularly due to the extreme wear engines experience during many military missions.

As has been illustrated, there is clearly a significant need for simulation in the development and maintenance of complex physical systems. While simulation is extremely important, engineers and scientists must be aware of the uncertainty associated with their models, and this should be a driving factor in selecting which models to use for a given analysis. These points will be further explained in subsequent sections of this proposal.

1.1 Modeling Complex Physical Systems

One common practice in modeling complex physical systems is to decompose the system into a series of less complex parts. Each part is commonly referred to as a Contributing Analysis (CA). Through this decomposition the system can often be better represented; the decomposed parts are often much better understood than the system as a whole. Therefore, each CA is analyzed individually, and the CA's performance is used to determine the system's performance. For example, gas turbine engines are used in the aerospace industry for thrust or power production on both fixed wing aircraft and rotorcraft, as well as for power generation in the energy business. The gas turbine engine operates using a Brayton cycle-based thermodynamic process.[144] Gas turbines are composed of physical parts such as a diffuser, a compressor, a burner, a turbine, and a nozzle. The gas turbine is a complex physical system whose individual parts are generally understood better than the system as a whole. To conduct analysis of gas turbines, it is common practice to decompose it into a series of parts and analyze each individually to calculate the entire system's performance.

Once the complex physical system has been decomposed into a series of CAs, it is common practice to link these CAs together in an integrated Modeling and Simulation

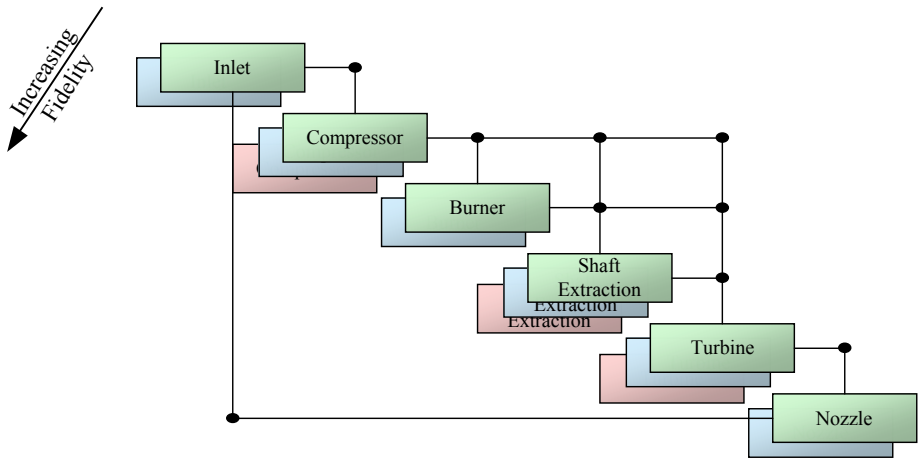


Figure 1: Example Multi-Fidelity M&S Environment.

(M&S) environment. This M&S environment is a computer environment where the CAs are able to communicate with each other and any physical or compatibility constraints or conditions between the CAs are handled by a system solver. Groups including but not limited to government, industry, and academia commonly conduct analysis using such environments and have done so for a number of years.

Over time, advances in computing and the understanding of complex physical systems have led to alternative solution methods. In many instances there are several different methods capable of performing analysis for a given CA. Because of the solution method and simplifying assumptions in each analysis tool, each has its own specific epistemic model uncertainty and runtime. One prevalent issue is that the process for selecting which tool to use to fulfill the need for a given CA is not well defined. If we were to consider all of the options we have for each CA, we would see multiple levels of fidelity for each CA. A notional example of these multiple levels for each CA in a gas turbine engine is provided in Figure 1.

When considering all of the analysis tool options, a user is faced with selecting which fidelity level they want for each CA in the environment. Currently, this selection is primarily ad-hoc, being based upon "practical" reasons such as users' trust, the heritage of the tool, or expert judgement for a specific case. This problem must be resolved, and it is the purpose of this study to create a rigorous traceable method for analysis tool selection based on the

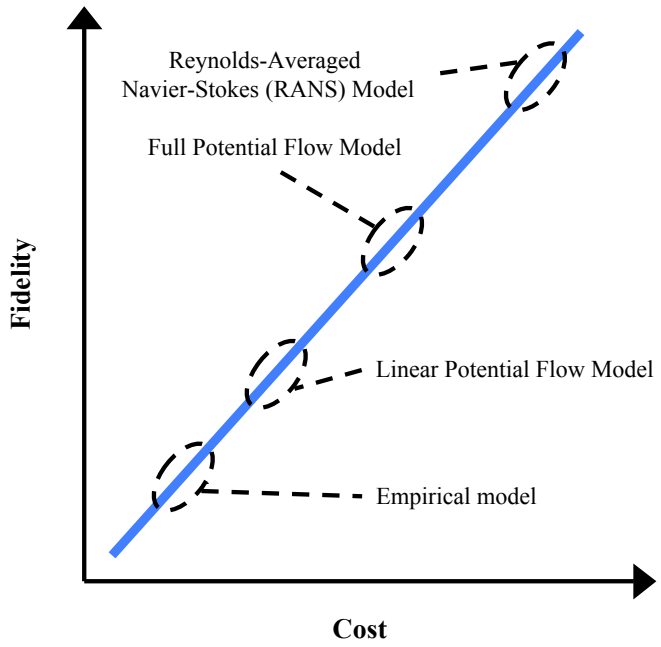


Figure 2: Notional Fidelity-Cost Trade in Aerodynamics Performance Model Calculation.

required fidelity of the M&S environment's outputs.

1.1.1 Accuracy, Fidelity, and Multi-Fidelity in an Modeling and Simulation Perspective for Complex Physical Systems

Before beginning a detailed discussion on modeling complex physical systems, one should first examine some key terms and definitions for the purpose of this study. Fidelity is defined to be "exact correspondence with fact or with a given quality, condition, or event; accuracy".[37] For the purposes of this document, accuracy and fidelity have the same meaning and will be used interchangeably, though it is acknowledged they have different meanings in some circles. Similarly, "multi-fidelity" in a M&S perspective refers to an M&S environment with multiple fidelity levels for a given CA, meaning there are different analysis tools to choose from. Fidelity and cost (or similarly accuracy and time) are positively correlated; this represents a fundamental trade in design. A notional example of the trade between fidelity and cost is shown in Figure 2 where four different methods of varying fidelity for calculating high temperature air properties are shown.

When selecting the fidelity level for an analysis, it is important to appropriately balance

the fundamental trade between cost and accuracy. Consider the following thought experiment: a user is posed with the question of selecting a fidelity level to perform a CA in the development program for a new complex physical system. If the user selects a fidelity level higher than what is needed for a given task, it would be a problem, possibly a very significant one. Referring to Figure 2, this would be analogous to choosing a full potential flow model when an empirical model was acceptable. This selection would tend to increase the cost and time required for a study. By making this poor decision, there would be an increased program cost because of the additional analysis development and execution; this could lead to the possibility of program cancellation. Conversely, if the user selects a fidelity level lower than that needed to complete the task there could also be an issue. In Figure 2, this would be similar to selecting a linear potential flow model when a full potential flow model is needed. If this mistake were made, there would be an unacceptable model uncertainty in the proposed solution, which may cause the program to pursue a "bad" design. If a "bad" design were selected because of the unacceptable model uncertainty, it would lead to increased program costs in the later phases of design, and possibly lead to program cancellation. Through this illustration, it should be clear that it is important for the user to select the appropriate fidelity level for a given task. By selecting the appropriate fidelity level, the user can minimize the amount of time required to do an analysis while also meeting a program's accuracy needs.

1.2 Fidelity Selection Current State of the Art and Perceived Use Cases

Fidelity selection is not an entirely new problem, though it is becoming more significant as computer power increases and alternative formulations continue to be formulated. This issue is addressed in several different ways. In some instances, "trusted" tools are selected regardless of their fidelity level. Another selection method involves always selecting the lowest fidelity level. The reasoning behind this approach is that while the answer may not be as accurate as possible, it will be cheaper than other analyses, and will therefore allow the user to explore more of the design space. However, as previously discussed, this could lead to selecting poor designs and jeopardizing the program's success. Another

possibility is to always select the highest fidelity tool combination. This would be analogous to someone selecting Computational Fluid Dynamics (CFD) and Finite Element Methods (FEM) analyses regardless of the information available for the problem or accuracy needed. Another method, proposed by Nixon, involves selecting analyses based on whether or not certain model effects are significant.[166] This method is similar to what is being proposed in that significant assumptions in the model are examined, but Nixon's approach to fidelity selection requires expert opinion for tool selection. A user would have to be familiar with the assumptions in all of the complex models they were using and also know when they were and were not valid. Another method used in designing printers and high-speed milling, among other applications involves using Pareto-optimality to make selections given some accuracy constraint.[127, 137] This is similar to selection methods from robotics and microchip design which use power and size constraints to select along theoretical Pareto frontiers.[16, 211] A more expert-driven method developed for use with transaction-level modeling involves the user subjectively identifying "interesting" portions of the analysis which are done in a higher fidelity while "uninteresting" portions are done much more quickly with lower fidelity tools.[20] While these methods may reveal analysis combinations which meet certain criteria posed by the user, a rigorous, traceable process is needed. To complicate matters, by re-examining Figure 2 the discrete nature of this selection problem eliminates many continuous optimization and selection methods commonly used. The previously outlined selection methods do not address key issues for the following use cases:

1. Analysis tool selection to provide necessary concept resolution.

2. Analysis tool selection when accuracy goals exist for several different outputs from the multi-fidelity M&S environment.

1.2.1 Analysis Tool Selection to Provide Necessary Concept Resolution

When evaluating different concepts with simulation tools, many first order studies are done deterministically. This is a serious problem because, as shown in Figure 3, by not considering the uncertainty, the results for the study may be negated. This is illustrated by comparing the deterministic results in the left-most pane to the same results in the center pane. The

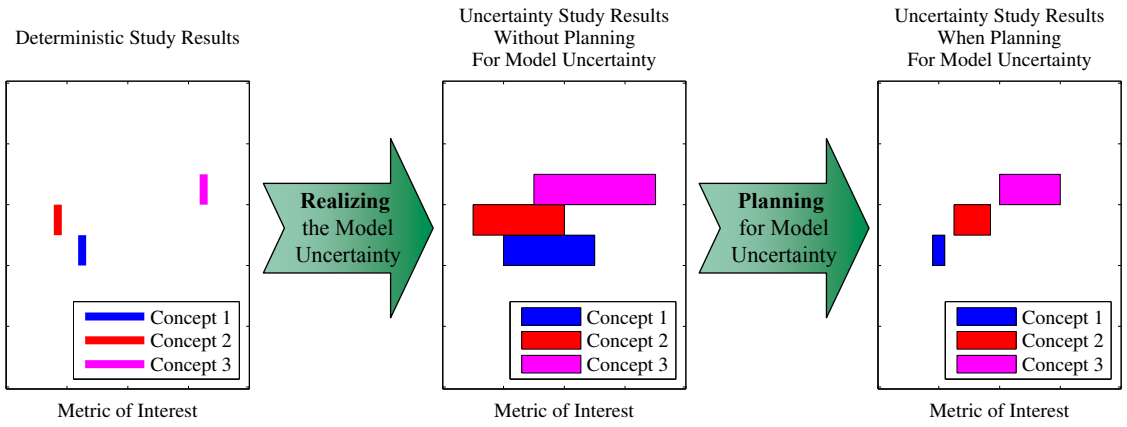


Figure 3: Notional Results for Studies with Increasing Attention to Model Uncertainty.

center pane represents the same study with the model uncertainty from the left-most pane quantified. This model uncertainty is represented by bands over a metric of interest. As can be seen in the center pane, the uncertainty bands for concepts 1, 2, and 3 overlap which means one can not differentiate the concepts from each other in the study. If the objective for this concept selection problem was to minimize the objective of interest, one would deterministically choose concept 2. However, by accounting for the model uncertainty that was present in the tools used in the left-most pane, a decision maker realized they are unable to differentiate between concepts 1, 2, or 3.

This could be avoided if the decision maker included the model uncertainty present in his/her analysis tools when selecting which ones to use. This third preferable scenario's results are shown in the right-most pane; here the decision maker could differentiate between concepts 1, 2, and 3. If the decision maker is trying to minimize the metric of interest, the decision maker would now make an informed decision which was predicted with a different set of analysis tools, and select concept 1.

A similar scenario could exist for technology trade studies or design reviews when the government or a company is trying to determine what configuration to pursue for an engine, power plant or aircraft. By not only recognizing that there is model uncertainty, but also planning for it when selecting the appropriate tools to conduct an analysis, the decision maker is able to make the correct, informed decision. Giving decision makers this capability

is a use case which is both important and cannot be rigorously completed given the current analysis selection capabilities.

1.2.2 Analysis Tool Selection When Accuracy Goals Exist For One or Several Outputs

As outlined earlier, there are several ad-hoc methods commonly used for analysis tool selection. However, if a method existed where the desired accuracy in metrics of interest could be specified and an algorithm would appropriately choose tools to meet this accuracy requirement, time and money could be saved. Users would no longer have to make ad-hoc tool selections and instead would have a traceable, rigorous method for choosing analysis tools for each CA. By giving the users this freedom, they would have the ability to reduce cycle time for an integrated analysis.

This could also be used to have the user focus on certain areas. In an aircraft mission example, a given design review could have accuracy requirements for system level metrics, such as the aircraft range, and the structures group may internally require a higher accuracy requirement for the FEM analysis. By having multiple accuracy requirements, users would be able to maintain system-level accuracy while also conducting usable higher fidelity work in their specialty.

If a fidelity selection algorithm were developed, an objective such as this could be achieved by appropriately specifying an objective function and constraints. In section 1.2.1, the objective would be to minimize run time and the constraints would be resolution between the different concepts. Here, the optimization objective function would also be to minimize run time while the constraint(s) would be the accuracy requirement(s) in one or several outputs.

1.3 Overview of Proposed Method

The method proposed in this document is intended to address the use cases discussed in subsections 1.2.1 and 1.2.2, but more generally to allow one to determine which analysis options should to be used to conduct an analysis such that the answer is “good enough” and is achieved as quickly as possible. An overview of this method is provided in the process

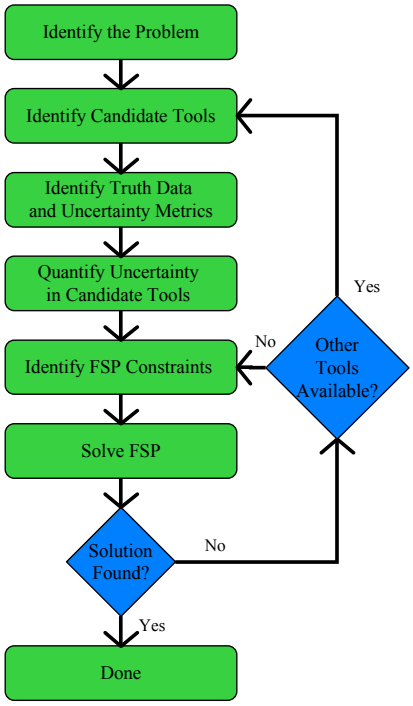


Figure 4: Process Flowchart for Proposed Method for Solving the Fidelity Selection Problem.

flowchart in Figure 4.

Step 1 - Identify the Problem. As with many processes, the first step in this process is to identify the problem. This could range from identifying analyses to perform a gas turbine performance analysis to identifying analyses to size the powerplant in an aircraft carrier.

Step 2 - Identify Candidate Tools. In this step, the possible tools which can be used for each CA should be identified. The candidate tools are dependent not only on the problem being solved, but also on the type of result determined. One must take caution and only select tools which are capable of producing the necessary results.

Step 3 - Identify Truth Data and Uncertainty Metrics. This step requires the users to find appropriate truth data. This truth data should be as representative of the system being modeled as possible. Once representative truth data has been obtained, possible uncertainty metrics should be identified. Ideally, one would quantify uncertainty in all metrics, but realistically, one should select only those most significant. When identifying

these metrics, sensitivity studies could be conducted to reduce the dimensionality of the problem. This will also serve to increase the speed for quantifying uncertainty in subsequent steps.

Step 4 - Quantify Uncertainty in Candidate Tools. When quantifying uncertainty in the candidate tools, a variety of approaches can be taken. This research assumes that reality is reality. Others may wish to assume that the highest fidelity tool available is "reality". If one takes the same approach as was taken for this research, the uncertainty quantification will be much more computationally expensive than the alternative approach, but the result is more conservative. It is the opinion of the author that it is more representative of the amount of information available to quantify model uncertainty in the various candidate tools.

If one were to assume reality is reality, then the uncertainty could be quantified using one of two possible approaches. If there are four or fewer input variables whose uncertainty was quantified in step 3, then the strategic distribution creation method, discussed in detail in Chapter 4, should be used. If there are five or more input variables, then the Frontier Finding Particle Swarm Optimization (FFPSO), discussed in detail in Chapter 6, should be used. The implementation details for each of these methods are discussed both in the respective chapters and later in Chapter 10.

Step 5 - Identify FSP Constraints. This step asks not only what "physics" is being represented in the model, but also how accurate the representation must be. For example, if one was designing a gas turbine engine, the expected life of the first stage High Pressure Turbine (HPT) bucket would be a possible constraint. If the study needed to predict this life within $\pm$ one thousand hours, that would be the uncertainty constraint.

Step 6 - Solve FSP. When solving the Fidelity Selection Problem (FSP), one should use the Genetic Algorithm (GA) developed in Chapter 8. This GA will determine which tool combinations best meet the provided objective function. This research was posed in such a way that the user is always requesting the tools to produce a "good enough" answer in the minimum amount of time, but it could be just as easily posed to request the best tool set given some maximum acceptable amount of time to perform computation.

Possible Iteration with Additional Tools. Once the GA has found a solution to the FSP in Step 6, some iteration may be necessary. If the constraint is too stringent, then it is possible there will not be a feasible solution to the FSP as it was posed. If this is the case, then one must determine whether additional tools exist which can be considered. If there are additional tools, these should be included. This may allow for a feasible solution to the FSP. If there are not, then the user should reconsider the constraints on the FSP, as they are too stringent, so that a feasible solution can be found.

This is intended only to provide an overview of the proposed method. For a more detailed discussion of its implementation, the user should consult the chapters referenced in the various steps, Chapter 9, or Chapter 10 for more information.

1.3.1 Proposed Method Applicability and Limitations

While this method can be used given the process outlined earlier, there are some limitations, filters, and such which must also be taken into account. With regard to identifying candidate analyses in Step 2, one cannot use a given analysis if it is unable to capture the metric of interest. For example, when considering gas turbine engine design, one can use a 0-D, 1-D, 2-D, or 3-D model to capture parameters such as net thrust and ram drag.

However, if one is interested in the spanwise velocity distribution at the nozzle exit, a 2-D or 3-D analysis must be chosen. Within each of these models, additional choices can be made which would represent adding or removing terms for a given dimension of analysis. For example, one may wish to see how various turbulence models or grid sizes would impact the uncertainty - time cost tradeoff. This is something that could also be considered with the method.

Compatibility among analyses is a significant concern when solving the FSP. Specifically, when integrating analyses to find combinations which may solve the FSP, the user must take care to ensure all combinations are compatible. Compatibility is not something that is specifically addressed in this document, but it is something the user must ensure when evaluating combinations.

When using the GA to solve the FSP, as is done in Chapter 8, one should ensure that

there are a sufficient number of combinations. If there are fewer than approximately 100 combinations, the GA will likely evaluate all possible combinations before converging, so one would be better off simply evaluating all combinations rather than using the GA to do so.

1.4 Overview of Thesis

In this chapter, a clear need for a multi-fidelity analysis selection method has been established. In Chapters 2 and 3, a better understanding of what is meant by uncertainty in general and more specifically epistemic model uncertainty will be provided, as well as specifics on how uncertainty will be propagated for this study. Implementation of this uncertainty propagation will be discussed in Chapter 4. An alternative method, using frontier-finding metaheuristic algorithms, will also be explored. The relevant literature is surveyed in Chapter 5. These methods will then be implemented in Chapter 6, where their performance relative to the initially intended approach covered in Chapter 4 will be compared.

After recommendations on how to quantify uncertainty have been made, the optimization aspect of the Fidelity Selection Problem will be examined. Relevant literature will be surveyed in Chapter 7 and the most promising methods' will be identified. In order to assess these methods' performance for the general case, a canonical form for the Fidelity Selection Problem was created in Chapter 8 and the promising methods performance was assessed. In order to better illustrate how this method can be implemented for an engineering problem, an illustrative example was selected and the uncertainty quantification method selected in Chapter 6 was used to quantify uncertainty in an Separate Flow Turbofan gas turbine engine. The fidelity selection was then performed for a series of notional scenarios and constraints using the discrete optimization method selected in Chapter 8. Finally, Chapter 11 begins with a summary of the work conducted for this study. This chapter also revisits the research questions, hypotheses, and experiments which guided this work, summarizes the author's contributions to the field, discusses some envisioned future work, and revisits key assumptions which guided this research.

CHAPTER II

SURVEY OF RELEVANT UNCERTAINTY THEORY

In this chapter, the theory associated with relevant uncertainty will be discussed in detail. Different taxonomies for uncertainty will be discussed and the kind of uncertainty that is the primary focus for this research will be identified. Theoretical representations for all relevant types of uncertainty will be discussed, traded and selected.

One should note that the purpose of this chapter is to identify a reasonable method for quantifying epistemic model uncertainty to quantify the differences between various tool options. The goal is not to develop a novel method for uncertainty propagation or quantification. Once a reasonable method for quantifying and propagating uncertainty has been identified, it will be used with discrete optimization methods to perform tool selection.

2.1 Reliability Engineering Uncertainty Perspective

Uncertainty is very generally defined as "the estimated amount or percentage by which an observed or calculated value may differ from the true value".[37] In the risk assessment and reliability engineering fields, a distinction between the types of uncertainty has been made for over 20 years.[57, 169] These two types of uncertainty are aleatory and epistemic uncertainty. Aleatory uncertainty is known to be the variation caused by inherent variability in a process. Aleatory uncertainty can be improved or reduced through process refinement and increased uniformity, similar to the 6σ process developed by Motorola. This is also known in various literature sources as irreducible uncertainty, stochastic uncertainty, inherent uncertainty, or variability.[101, 146, 168, 169] The second type is called epistemic uncertainty. This uncertainty is not caused by random variation in the analysis, but rather is caused by a lack of knowledge about the system. This lack of knowledge can be improved by gaining more understanding through a more in-depth model of the problem's physics or a more thorough experimentation. Epistemic uncertainty is also referred to as subjective uncertainty, reducible uncertainty, cognitive uncertainty, or bias in statistical and reliability

engineering literature.[101, 146, 168, 169, 207]

From an engineering statistics perspective, error can be defined as "unexplained variation in a collection of observations".[161] Oberkampf[167, 168] and Alvin[5] among others, have made a distinction between error sources. They break error into different parts: acknowledged error and unacknowledged error. Their definition is that acknowledged error can be attributed to some understood and realized difference between the model and reality, and to improve acknowledged error, alternative models should be consulted when possible. Unacknowledged error is said to be caused by coding mistakes and can be alleviated through code debugging and verification. Neither acknowledged nor unacknowledged errors will not be considered for this study. It will be assumed that any acknowledged error will be accounted for through corrections to the simulation or testing apparatus and that unacknowledged errors have been eliminated through an appropriate Verification and Validation (V&V) process.

Aleatory and epistemic uncertainties are both present and significant in conceptual design. Aleatory uncertainty is most commonly represented with probability theory and has been represented this way as early as 1908.[207] This is widely accepted by the reliability engineering community as the best alternative, and as such, it will be the only method examined for this study. Aleatory uncertainty will be represented with a normal distribution whose mean and standard deviation are known based on V&V processes currently in place, though this method could be implemented for an arbitrary distribution.

Epistemic uncertainty was initially modeled with probability theory. Users would often take a Principle of Insufficient Reason approach when selecting probability theory for these "most uncertainty" of outcomes. This principle gives all outcomes the same probability, and in a continuous space, it would be represented as a uniform distribution. However, using this approach when no or very little information is available will intrinsically assume that the user knows more about the problem that they do. In the 1980's, the uncertainty propagation and representation community decided it was a poor method to represent epistemic uncertainty.[122] The main argument against probability theory is that imprecision

and ambiguity or vagueness can not be represented. For these issues, four alternative methods will be outlined and compared to decide which would be best suited for representing epistemic uncertainty in a situation where little information is known about the problem.

2.1.1 Epistemic Uncertainty Theoretical Representation

As outlined earlier, probability theory has been identified by many in the risk analysis and uncertainty propagation communities as a poor method for representing epistemic uncertainty because of the theory's inability to represent vagueness and imprecision.[13, 103, 109, 123, 122] There are many advantages to using probability theory for uncertainty propagation, such as over 100 years of experience, a gamut of different propagation methods, and a clear, working understanding of how the theory works. However, its inability to represent vagueness and imprecision means it is not suitable for epistemic uncertainty propagation. This section will outline several alternative theories suitable for quantifying epistemic uncertainty, specifically Bayesian theory, possibility theory, and evidence theory. Following an overview of these theories, each will be compared and the most promising theory will be identified.

2.1.1.1 Bayesian Theory

Bayesian theory is a possible method for epistemic uncertainty propagation and is essentially an extension of probability theory. Bayesian theory is named after Thomas Bayes, an English clergyman. Bayesian theory involves assigning a prior distribution to a given event, and then updating this prior distribution as more information is gained to more accurately reflect reality.[9] This is formally represented with Bayes' Theorem in Equation 1.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

Where:

- $P(A)$ is the prior probability
- $P(A|B)$ is the posterior probability of A, given B
- $P(B|A)$ is the posterior probability of B, given A

- $P(B)$ is the prior probability of B

As stated earlier, this mathematically provides a method for updating the probability of A occurring (to be more representative of reality) given that B (an event or experiment) has occurred. This process can be iterated as data is made available, allowing one to begin with a very rough estimate of uncertainty and arrive at a much more refined estimate. Given several updates, even a poor estimate of uncertainty can be mitigated. The process commonly begins with uniform distributions, derived from Laplace Principle of Insufficient Reason[1], and as information is obtained, this prior estimate is appropriately updated.

Mantis developed a method in 2002 that uses a Bayesian statistics approach to accurately and effectively perform uncertainty propagation and update Probability Density Functions (PDFs) representing uncertainty in the early phases of conceptual design.[142] Many others have used Bayesian theory for object recognition for systems such as notional automated air traffic towers, surveillance systems, and data fusion applications, among others.[25, 106]

Because of its relative popularity in engineering applications, some confusion exists around Bayesian theory. Strictly speaking, it is a theory named after Thomas Bayes used for updating probabilities as more information becomes available. However, the term "Bayesian" has also come to represent a subjectivist approach to probability theory where the shape of a given PDF is assigned based on intuition of the user, who subjectively assigns the shape. The opposite of a subjectivist approach is a frequentist approach. A frequentist assigns the shape of a PDF based on forming a histogram with available samples. A subjectivist approach is a common practice in sports gambling, where odds are made and some would argue no prior information is available. A subjectivist approach for gambling does make sense, because the two teams or groups over which a notional wager could be made have never played each other in the same location with all of the same players under the same conditions, so no information is available to take a frequentist approach.

Bayesian theory is a very promising method to use for updating information through

[1]Laplace Principle of Insufficient Reason states that if no additional information is known about n possible outcomes, then the probability of each occurring is $\frac{1}{n}$. For a fair coin flip, this means there is a 50% chance of having either heads or tails. For a continuous problem, this represents a uniform distribution.

a design process, given enough information. However, for the application in mind for this study, there are two major problems. First, the method's roots in probability theory prohibit it from being able to characterize ambiguity or vagueness due to a lack of knowledge. In probability theory, the principle of additivity requires that the sum of the probability of an event occurring and the probability of the complement of an event occurring to be 1. Therefore, in using probability theory and Bayesian theory, one is unable to adequately state a lack of knowledge. Consider an example posed by Shafer in his 1976 work. When assessing the odds of intelligent life on another planet a Bayesian theory approach with no available information would default to the principle of insufficient reason. Here equal probability is given to both there being intelligent life and there not being intelligent life. However, this means there is a 50% chance of life on this distant planet and making a statement such as that is not something most on this planet are willing to make. The second issue with a Bayesian theory approach involves data used to update prior distributions and form posterior distributions. For the problem at hand, no additional information will be gained, so prior distributions cannot be updated. When selecting the appropriate tools for an analysis, additional experiments are not being performed to change the uncertainty associated with a given tool, so no new information can be incorporated. A Bayesian theory approach may make sense to quantify the change in uncertainty as one progresses from pre-conceptual to conceptual to detailed design, but it does not make sense for assessing the uncertainty in pre-conceptual, conceptual, or detailed design.

2.1.1.2 Possibility Theory

Possibility theory is a theory initially advanced by Zadeh with his 1978 article on fuzzy sets.[230] Zadeh's work was continued by others, primarily Dubois and Prade with their 1986 book on possibility theory.[65] Zadeh, Dubois, and Prade were proposing an alternative theory that is similar, though different, to probability theory for uncertainty propagation. Possibility theory is based on fuzzy measures, an extension of classical measure theory. When examining probability theory in a discrete sense, it could be said that each set is a singleton of a universal set, so that for the universal set, $\{X\}$, containing a, b, and c, there

could be the following three sets: $\{a\}$, $\{b\}$, and $\{c\}$. This differs from possibility theory, which represents discrete sets as part of a nested set, or chain, for a universal set. For the same universal set, $\{X\}$, there could be set $\{a\} \subset \{a, b\} \subset \{a, b, c\}$. This example will be continued in the subsequent section on evidence theory.

In probability theory, the probability of an event is what is calculated. In possibility theory, there is a different objective: to calculate the possibility of an event and the necessity of an event. There are several different meanings for the possibility and necessity which have been advocated by possibility theory's proponents. The first perspective was advanced by Zadeh in his 1978 work based on fuzzy sets.[230] Klir and Wierman proposed an alternative viewpoint, that the necessity and possibility represent the limit of the belief and plausibility, respectively.[123] The belief and plausibility of an event will be discussed later, as they are a part of evidence theory. The third perspective is that they represent the upper and lower probability of a system, which has been advocated by several researchers.[13] The most common of these is Zadeh's perspective, but it is useful to see that there is not a consensus even among the advocates for possibility theory as to what its most basic definitions are.

When examining operations in possibility theory, it is easiest to compare them relative to probability theory to highlight the differences. In probability theory, the sum of the probability of an event occurring and the probability of its complement occurring must be 1.0, as shown in Equation 2.

$$P(A) + P(\bar{A}) = 1.0 \tag{2}$$

However, this does not hold in possibility theory. Rather, the possibility that an event could occur is less restrictive than the probability that it could occur, and is therefore super-additive. This means that the sum of the possibility of an event occurring and the possibility of its complement occurring must be at least 1.0, as shown in Equation 3.

$$\Pi(A) + \Pi(\bar{A}) \geq 1.0 \tag{3}$$

Similarly, the necessity is sub-additive, meaning the sum of the necessity of an event

occurring and the necessity of its complement is no more than 1.0, shown formally in Equation 4.

$$N(A) + N(\bar{A}) \leq 1.0 \tag{4}$$

The sub- and super-additivity of possibility theory allows it to represent vagueness or imprecision about an event. Paraphrasing an example from Shafer's *Theory of Evidence*, consider the example of life on another world to illustrate the importance of vagueness or imprecision. If this event were characterized with possibility theory, one could more conservatively say that there may or may not be life on another world without saying that it is just as likely there is life on another world as there is not life. Mathematically, that could be represented as shown as $\Pi(Life) = 0, N(Life) = 1$ for the possibility and necessity, respectively.

Dubois and Prade also proposed a Bayesian-like method for combining sources of uncertainty. This has met some criticism in the literature due to its restrictions and inability to combine expert opinions.[13, 222]

While this is a weakness of a part of the theory, it does not necessarily mean the method is completely invalid. The intended theoretic application, representing epistemic uncertainty in a given design phase, does not require the combination of sources of uncertainty, so while this is a clear issue with possibility theory in general, it is not an issue for this intended application. There have been several studies which have successfully implemented possibility theory, including catastrophic failure identification, data fusion, and object recognition studies, among others.[25, 82, 165] It should be noted that while possibility theory was successfully implemented, it required more time to reach an answer than a Bayesian approach, and in the Nikolaidis study, possibility theory produces a significantly different answer. Nikolaidis goes on to say that because of this different answer, a probabilistic approach is more useful in later phases of design when a frequentist approach to probability is possible, but that in early phases of design when this information is not available, a subjectivist approach to probability should be paired with possibility theory to illustrate the variety of potential uncertain outcomes to the designer.

In addition to producing different results, one should note that the calculus behind possibility theory and probability theory is fundamentally different. Therefore it would be difficult to pair a possibilistic representation of epistemic uncertainty with a probabilistic representation of aleatory uncertainty. Some, including Klir[123], have advocated that there is a transformation from probability theory to possibility theory and then back, meaning it would be possible to combine probabilistic aleatory uncertainty with possibilistic epistemic uncertainty; the author feels this is unlikely. This sentiment is supported by others in the field[13, 165], and because a solution to the problem at hand is a necessity, using this approach to quantify epistemic uncertainty is not a possibility for the purposes of this study.

2.1.1.3 Evidence Theory

Evidence theory, also known as Dempster-Shafer theory or Dempster-Shafer Theory of Evidence, is another alternative to a subjectivist approach to uncertainty propagation. The concept was initially proposed by Dempster in his 1967 paper on upper and lower probabilities.[51] This concept was furthered by Shafer with his 1976 book [190] where the concept of upper and lower probabilities was extended to the plausibility and belief of an event. There are three accepted approaches for using evidence theory's belief and plausibility metrics: a random set, a generalization of the Bayesian approach, and an upper and lower probability.[195] While the second may be most like what Shafer intended, the third is more closely aligned with Dempster's work and is the approach being considered here. This approach stems from R.A. Fisher's fiducial inference. With this approach, Fisher showed that statistical inferences can be drawn without making prior distribution assumptions, as is done in Bayesian theory. This is likely why there is some confusion between a subjectivist and a Bayesian approach to probability theory. A frequentist would not be able to make prior distribution assumptions, therefore requiring a fiducial inference. Subjectivists, however, would have no problem with this and would pursue a Bayesian theory approach.

If one were to reject a subjectivist approach to probability theory, evidence theory could

be thought of as a method which performs well when there is not enough information available for a frequentist probabilistic approach. Evidence theory, unlike possibility theory, is compatible with a probabilistic approach; this comparison can be made as more information is acquired, allowing one to update belief and plausibility estimates.

When examining evidence theory for a discrete set of data within universal set $\{X\}$, containing a, b, and c, the singleton set limitation from probability theory does not apply. The possibility set limitation of nested sets does not apply either. An evidence theory representation of universal set $\{X\}$ could have sets $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, and $\{a, b, c\}$. Comparing these possible sets to those which could be captured with probability or possibility theory, one can see that evidence theory is able to much more diversely represent a dataset.

Looking at the basic operation of evidence theory, similar sub- and super-additive properties are seen as was the case in possibility theory. The sum of the belief of an event and the belief of its complement are said to be less than or equal to 1.0, as provided in Equation 5. The plausibility, similar to the possibility discussed earlier, is super- additive, meaning the sum of the plausibility of an event and the plausibility of its complement is at least 1.0, shown in Equation 6.

$$Bel(A) + Bel(\bar{A}) \geq 1.0 \tag{5}$$

$$Pl(A) + Pl(\bar{A}) \leq 1.0 \tag{6}$$

There also exists a relationship between plausibility and the complement of an event, as shown in Equation 7

$$Pl(A) = 1 - Bel(\bar{A}) \tag{7}$$

There are two different perspectives on belief and plausibility which will be discussed. The first is Dempster's perspective that belief and plausibility represent the lower and upper probabilities of an event, respectively. The second way is Shafer's perspective. He

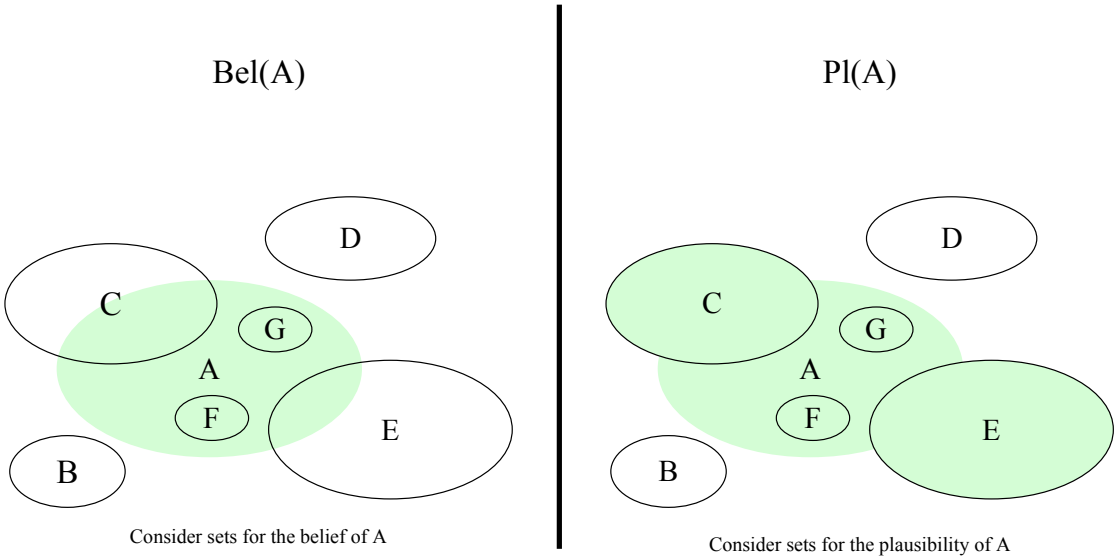


Figure 5: Graphical Representation of Belief and Plausibility.[82]

stated that belief contains the evidence which supports an event, while plausibility contains the evidence which does not oppose it. While this abstractly may not be clear, consider the graphical representation in Figure 5. Here the belief of A represents only elements completely contained within A. These are elements A, F, and G. Elements C and E contain some overlap with A, but are not completely within A. Therefore, they are not part of the belief of A. When looking at the plausibility of A, this represents any elements which contain at least part of A. This includes elements A, G, F, C, and E.

As was the case with possibility theory, evidence theory has the ability to represent a lack of knowledge. Shafer illustrated this in his 1976 work on evidence theory with the life on another world example. While many may hope there is life on another world, it would be unrealistic to say there is a 50% chance that there is life on another world. If set L corresponds to there being life on another world and set $\bar{L}$ corresponding to there not being life on another world, this lack of knowledge could be quantified by saying $Bel(L) = 0$ and $Bel(\bar{L}) = 0$, whereas probability theory would require $P(L) = 0.5$ and $P(\bar{L}) = 0.5$ according to the Principle of Insufficient Reason.

Now that the basic operations for evidence theory and how discrete sets are represented within this theory have been provided, one could show that, given certain limitations,

evidence theory can be used to represent probability theory and that evidence theory can similarly be used to represent possibility theory. Re-examining the possible sets for universal set $\{X\}$, if one were to exclude all non-singleton sets and if one had enough evidence to state that the sum of the belief of an event and the belief of the complement of an event was not less than 1 but actually equal to 1, then one would have a probabilistic representation of this evidence. Similarly, if one were to only consider nested sets in universal set $\{X\}$, one would have a possibility theory result. This has been mathematically proven by many, but this derivation will not be included. For more information on this topic, please consult Klir or Shafer's works.[123, 190] It is also interesting to note that in having seen that significantly different derivations are needed to get from evidence to probability and evidence to possibility, the likelihood of being able to move from possibility to probability is quite low.

As just illustrated, probability theory can be shown as a special case for evidence theory. Therefore, one would expect there to be a more general form of Bayes Theorem, which was provided in Equation 1. This generalized form of Bayes Theorem is called Dempster's Rule for Combination. While it is very similar to Bayes theorem, it has received considerable criticism for its inability to account for certain situations where conflicting, non-independent, or only somewhat reliable information is provided.[222] While this is an obvious issue with Dempster's Rule for Combination, it is a separate issue from whether or not evidence theory is an acceptable method for uncertainty representation. Additionally, given that the intended implementation will not involve any uncertainty combination (though Dempster's Rule, Bayes Theorem, or other methods), the inability of Dempster's Rule for Combination to account for these situations should not be considered when selecting an appropriate method for this application.

Evidence theory has seen significantly more acceptance by the reliability engineering and uncertainty community relative to possibility theory. Evidence theory has been used in a variety of approaches, from sampling-based representations of epistemic parameter uncertainty in nuclear reactors [102] to artificial intelligence object recognitions algorithms[106] to medical imaging analysis[28] to name a few. While it has been used considerably more

than possibility theory, it is not nearly as popular or accepted as a subjectivist/Bayesian approach, though many have begun to realize the weakness in a subjectivist approach to uncertainty and have re-examined a fiducial inference approach.[205]

While evidence theory does show promise, there are two issues stated in the literature with using evidence theory for uncertainty representation:

1. There is a lack of "operational" understanding of what belief and plausibility represent.[13]
2. There is a significant computational expense associated with performing an evidence theory implementation.[102]

These are significant issues which must be addressed if one were to implement an evidence theory approach.

2.1.1.4 Epistemic Uncertainty Representation Conclusions

Three different epistemic uncertainty representations have been examined: Bayesian theory, possibility theory, and evidence theory. Of these three, each has issues which require resolution, but some are more promising than others. In order to compare these methods, four criteria were identified which will impact a candidate method's effectiveness with quantifying epistemic uncertainty for this problem.

1. Applicability to this problem.
2. Method clarity in theory and implementation by others.
3. Acceptance by the reliability engineering community.
4. Sparse data applicability.

Applicability to the problem, specifically quantifying uncertainty associated how a code's representation of reality differs from reality, was used as a criterion for comparison and is intended to represent the method's compatibility with a problem. Specifically, a small amount of information is available to quantify uncertainty, and this information will not change or increase, meaning new information will not become available midway through a

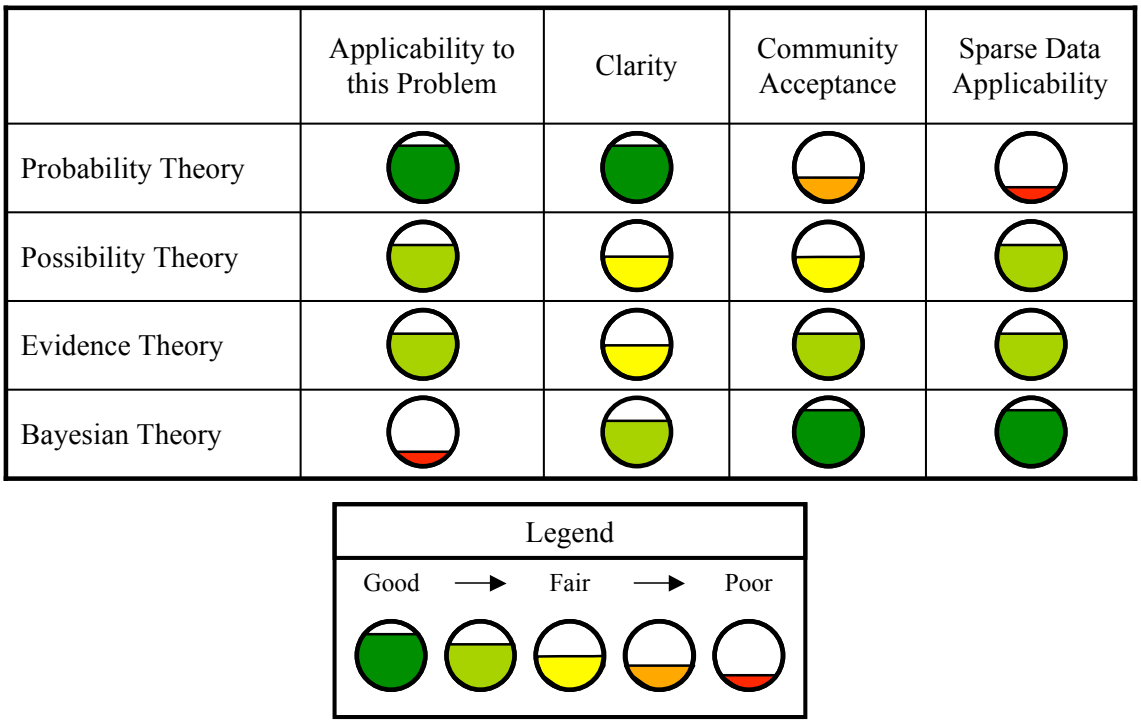


Figure 6: Qualitative Comparison of Uncertainty Propagation Methods.

process. The second criterion, method clarity, is intended to capture how straightforward the method would be from both a theoretic and implementation perspective. While some may consider this a poor metric for comparison, nondeterministic analysis is in no way a simple process, and even if a method was able to perfectly capture uncertainty for this problem, if the method was extremely complex and confusing, the likelihood of it being accepted by users and correctly used would be low. The third criterion is reliability engineering community acceptance. This captures whether or not other reliability experts agree that the method is valid for epistemic uncertainty quantification. This chapter is intended to survey possible methods and identify the most promising for implementation. If one were to select a method which has been rejected by most of the community, the findings from resulting studies using this questionable method would also likely be rejected. The final criterion is applicability to a sparse data problem. The problem at hand involves sparse data, and therefore, barring a subjectivist approach to the problem, certain methods must be eliminated from comparison.

This comparison is provided in Figure 6. One should note that while probability theory has been included in the comparison, it is done primarily as a benchmark for the other methods. Probability theory scores well for applicability to the problem because it works well for quantifying uncertainty if sufficient information is available for a given point in the design process. It also scores well for clarity because most engineers are not taught non-deterministic analysis, but if they are, it is nearly always a probabilistic approach. Therefore, it will act as the "gold standard" to compare other methods against. Moving on to acceptance by the community, one can see that probability theory scores "fair". This is in light of the argument presented by the reliability community in the late 1980's that probability theory implicitly assumes more than is often known about a problem. This is because there is often sparse data when assessing epistemic uncertainty. Unless one was to take a subjectivist approach, probability theory would be a poor choice because the subjective choice of distribution type and range would falsely appear as certainty in the modeling uncertainty of a result.

Possibility theory, evidence theory, and Bayesian theory all work well with sparse data. Possibility and evidence theory were developed for the purpose of being able to assess vagueness and imprecision, and Bayesian theory is commonly used to solve sparse data problems where the subjective distributions are updated as more information becomes available. With regard to acceptance by the community, theoretical flaws in possibility theory have been pointed out, so it only performs moderately well for this metric. Evidence theory is gaining acceptance, primarily in the object recognition community, and Bayesian theory is commonly used and accepted by the reliability engineering community, if the user is willing to take a subjectivist perspective and has sufficient data to update their Bayesian distributions a few times. With respect to clarity, the three alternatives score approximately the same, each having some similarities to a probabilistic approach. Some confusion may exist when users try to assess how the nested set implementation for possibility theory works. There is also some confusion in what implementation is most appropriate (upper and lower probabilities, Bayesian generalization, or random set) for evidence theory. Finally, when assessing applicability to the problem both possibility and evidence theory score relatively

well. However, Bayesian theory scores poorly because its key difference from probability theory, and therefore acceptance by the community, is the distribution updating. Bayesians make subjective assumptions about prior distributions, but these are then updated as information becomes available. For this implementation, no new information will become available. If one was to use Bayesian theory to quantify uncertainty and one was moving from a pre-conceptual to a conceptual to a preliminary to a detailed design, this would be a good approach. However, since this is not the problem being solved, Bayesian theory does not significantly differ from probability theory for this application.

Based on this qualitative assessment, **the most promising candidate for representing epistemic uncertainty in this study is evidence theory.**

Because certain issues were identified with using an evidence theory approach for epistemic uncertainty quantification, this brings forth two additional issues which must be addressed, specifically an appropriate operational definition for belief and plausibility for an analysis selection problem and the computational expense associated with propagating the uncertainty. These issues will be addressed and methods for resolving these issues will be compared and identified.

2.2 Modeling and Simulation Uncertainty Perspective

A Modeling and Simulation (M&S) perspective does not need to identify the root cause of the uncertainty but is concerned with how to model it. Therefore, from an M&S perspective, there are two distinct types of uncertainty: parameter uncertainty and model uncertainty. Parameter uncertainty is as it might sound: uncertainty in what the value for a specific parameter might be. This has been examined by many, including the reliability engineers in models of nuclear power plant waste, among other applications.[103]

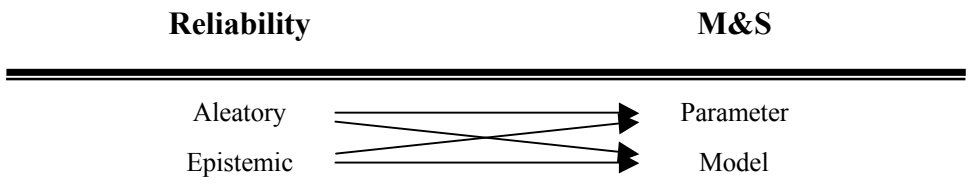
Other instances of parameter uncertainty include modeling technologies and their impact on an overall system. For example, when a technology is implemented, how much improvement can truly be gained? If modeling an engine, “how much will the Overall Pressure Ratio (OPR) increase” or “how much can the turbine inlet temperature increase” are common questions asked by gas turbine designers.

The second kind of M&S uncertainty is model (or code structure) uncertainty. This uncertainty is similar to code verification in that it asks the question "how close to the 'correct' equations are the equations that I am evaluating?" While, in the most correct sense, one should be conducting a non-equilibrium reacting gas calculation at all times, depending on what the flow properties are, one could assume a Calorically Perfect Gas (CPG) and achieve an answer very close to the correct one in orders of magnitude less time. Model uncertainty has been modeled primarily with Bayesian networks [23, 218] where probabilities are placed on the interactions of different nodes, and Bayes factors [63, 116, 139], an approach where two models are compared based on their ability to represent a series of data. However, these schemes are only of use if assumptions are made and the distribution shapes can be updated in a Bayesian sense. Because more information is not being incorporated into the model to improve how likely a calculation is being done in a specific way, these methods are not of use. One alternative, published by Du & Chen in 1999[64] and later adopted by Gu et. al.[84] , involves applying "adder" functions to results of a code in a probabilistic approach. This approach could be implemented in a non-Bayesian method and therefore is of interest for the purposes of this study.

2.3 Mapping Reliability and Modeling and Simulation Uncertainty Perspectives

While a reliability engineering categorization of uncertainty and an M&S categorization of uncertainty are different, they are in many ways similar, and a mapping, provided in Figure 7, can be constructed between the two approaches. In literature, there exist four different mappings: aleatory parameter uncertainty, epistemic parameter uncertainty, aleatory model uncertainty, and epistemic model uncertainty.

Aleatory parameter uncertainty is the most common type of uncertainty examined and is commonly what people think of when the topic of "uncertainty propagation" is brought up. An example of this type is the random variation in a component dimension due to manufacturing process variations. This type of uncertainty is represented with probability theory by most of the engineering community, and as such is the representation that will be used for this study. The second uncertainty type from this reliability/M&S mapping



	Uncertainty Type	Theoretic Representation
1.	Aleatory parameter uncertainty	Probability Theory
2.	Epistemic parameter uncertainty	Evidence Theory
3.	Aelatory model uncertainty	Probability Theory
4.	**Epistemic model uncertainty**	**Evidence Theory**

Figure 7: Reliability and M&S Uncertainty Mapping.

is epistemic parameter uncertainty. This, though less common than aleatory parameter uncertainty, is the type of uncertainty being considered when others address "epistemic uncertainty propagation". An example of this type of uncertainty would be the results from many of the Sandia National Laboratory studies of epistemic uncertainty for nuclear waste. Here, the physics of the problem have been accepted, but the specific values of a given parameter may not be known. Epistemic parameter uncertainty will be modeled in this study using evidence theory. The third type of uncertainty, aleatory model uncertainty, is commonly examined in statistical mechanics oriented studies. Here there is fundamentally a random variation associated with the model. This type of uncertainty has been considered for completeness but is not relevant to the study at hand. The fourth and least commonly seen type of "uncertainty propagation" is epistemic model uncertainty. Here the underlying physics of a model are unknown because of a lack of knowledge. This type of uncertainty is used to quantify how "close" the physics of the problem may be to some reference and is the type of uncertainty which will be the focus of this research. This type of uncertainty will be modeled using evidence theory.

When modeling epistemic model uncertainty with a non-Bayesian approach, some development is needed. A promising method was illustrated by Du & Chen as well as Gu et. al. to represent model uncertainty as an "adder" function added to the result of interest.[64, 84]

Du & Chen proposed using a probability theory approach where PDFs would be added to results of certain codes based on information available. However, as discussed earlier, using probability theory to represent epistemic uncertainty is a poor tactic. Gu et. al. took a different approach, using a "worst-case"interval analysis approach. Based on this information, **the author proposes using this "adder" approach to capture epistemic model uncertainty for the Fidelity Selection Problem (FSP).**

A notional representation of the proposed uncertainty modeling approach is provided in Figure 8. Here a series of known inputs are passed to a tool. The analysis being performed is known, but in order to account for the model uncertainty, an "adder" distribution is added to each output where uncertainty information can be quantified. Because an evidence theory approach is being proposed, the first of the previously identified issues in an evidence theory implementation is that there is a lack of operational understanding of what belief and plausibility represent, which must be addressed.

The "adder" approach intrinsically addresses the operational definition issue, which was shown to be a potential limitation of evidence theory. The belief, as defined by Shafer, is the result that available evidence shows to be the answer. While this is the definition traditionally used, a consensus has not been reached in literature as to how this should be implemented. For this research, our goal is to establish an upper and lower bound for the uncertainty in a given model output, and Shafer's general definitions for plausibility and belief can be used as a vehicle to obtain these bounds. The belief can be obtained from V&V data and would be the value furthest from the analysis result. The plausibility, which is a value that has not been disproved by data, would be the bound closer to the codes result but must contain the codes result. Consider a few examples to illustrate this implementation. If only one datum point was available, it would always be the belief and the analysis codes result would always be the plausibility. If two data points were available, one at 10% more than the code's result and one 5% more than the code's result, then the belief would be +10% (the value furthest from the code's result) and the plausibility would be 0% (which is not the lower bound from the data, but is established by the code's result). If these data points were instead at 10% higher and 2% lower than the code's result, the

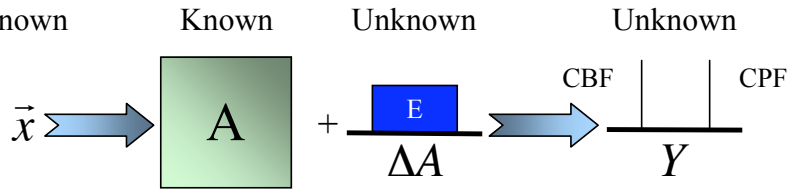


Figure 8: Proposed Approach for Modeling Epistemic Model Uncertainty

belief would be +10% and the plausibility would be −2% (the lower bound, because this scenario's bounds contain the codes result). A third scenario, where one data point is 10% higher and one is 10% lower, then the belief and plausibility would be interchangeable at ±10%. The example could easily be extended if n points were available. Once these upper and lower bounds for the "adder" are obtained, they can be incorporated into the model as illustrated in Figure 8, which is very similar to that used by Gu et. al. Here, given a known input (x) to a known analysis (A), some unknown "adder" (ΔA) is added to the result. This produces an unknown range of outputs (Y), which vary between some bounds. This differs from an interval analysis because of the nondeterministic aspect of the evaluation. In an interval analysis, the high and low for the "adder" would be evaluated and then propagated to the result, Y. In a probability theory approach, the results would be interpreted as some probability of an event occurring, a 10% probability of some outcome. Evidence theory takes a more general approach. Here the results should be interpreted by the upper and lower bound for the space, or Cumulative Belief Function (CBF) and Cumulative Plausibility Function (CPF), respectively. This approach would perform a similar process, but rather than only evaluating the high or low value in the "adder", it considers all possible settings with all possible outputs of analysis A.

An illustration of how this would be implemented may clear this question. Consider the following example: an analysis code's "adder' is to be determined. One of the outputs of this code is γ, the ratio of specific heats ($\gamma = \frac{C_P}{C_V}$).[113] These results are being used later in another analysis code to calculate information about a high temperature flow.

The actual variation of γ is known to be a function of pressure and temperature, but for this example, assume this information is not known. Instead, there are two experimental samples available which were taken under conditions similar to those the analysis code will

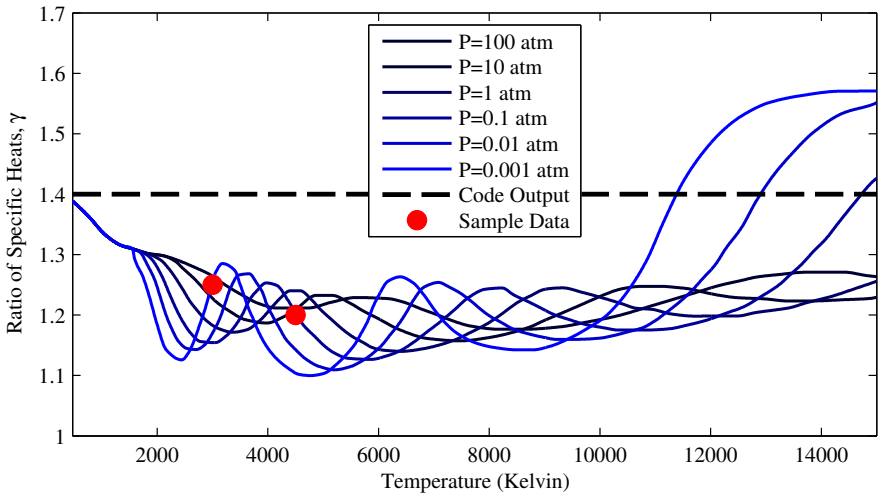


Figure 9: Variation of γ with Pressure and Temperature.[6]

predict. These samples indicate there is a variation in γ, but this is insufficient data to establish realistic trend lines. The actual variation of γ can be seen along with the output of the notional analysis code and the two data points in Figure 9.

It should be noted that if there is not enough information available to capture this variation, this correction should be incorporated into the model, but for this example there clearly is not. Therefore, this "adder" approach will assign a distribution varying the code's output, γ. The two samples available, one predicting $\gamma = 1.2$ and the other predicting $\gamma = 1.25$, can be compared against the analysis code's result of 1.4. The first sample, $\gamma = 1.2$, is 14% below the code's predicted value, and the second sample is 11% below the code's predicted value. From this, the "adder" ranges can be assigned. For this example, the "adder" will range from 0% (possibility of the code being correct) to 14% (sample furthest from code's prediction), which corresponds to a distribution covering the variability in γ that was determined from our two data points.

Some may consider this to be a strong statement, but it is the author's opinion that this is acceptable. If additional uncertainty information were available, perhaps a more in-depth approach which maps the uncertainty as a function of the input parameters could be adopted, but this approach was all that the author felt should be made given the sparsity of data available.

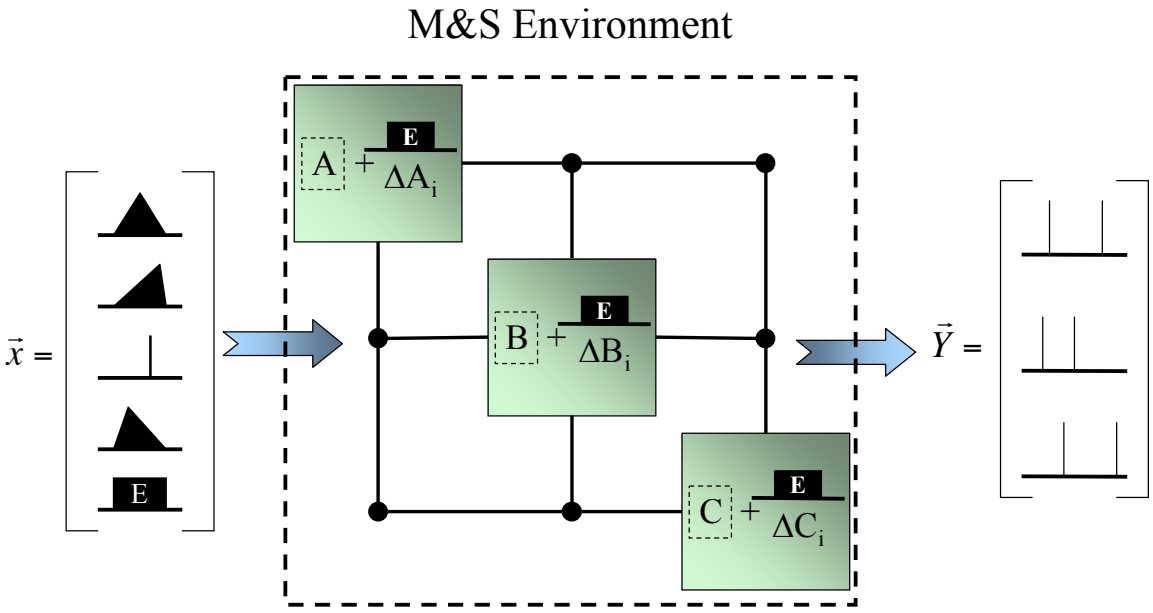


Figure 10: Proposed "Complete" Uncertainty Representation

When implemented, this "adder" approach should be applied to any variables where information is available to calculate these parameters.

2.4 "Complete" Uncertainty Representation

While the goal of this study is to quantify epistemic model uncertainty, this process could also be used for the other types of uncertainty created from the reliability/M&S mapping provided in Figure 7. A notional representation of this is given in Figure 10.

Using this approach, a vector of global inputs is passed into the integrated M&S environment. These inputs can be fixed values in a deterministic study (represented with vertical lines at a fixed value), aleatory parameter uncertainty (represented with various distributions corresponding to the random nature of the variable), or epistemic parameter uncertainty (represented with uniform distributions with an E in them). These inputs are fixed for a

given study. Therefore, if one were to examine, for example, thermodynamic cycle changes for a given engine, these input parameters ($x's$) would be varied. Looking at the environment itself, one can see the individual Contributing Analysiss (CAs) (A, B, and C). Encapsulated just outside of each CA is the "adder" function applied to each

CA output with available uncertainty quantification information. While the global input uncertainty is specific to the problem being examined, the "adder" functions have each been calculated based on a V&V process and are specific to the class of problem for which the V&V process was conducted and to the code itself. When examining the multi-fidelity aspect of a given problem, the user is in essence swapping out each block (the CA and its associated '"adder" function) when changing from one fidelity level to another. Finally, the outputs from the environment are not a series of Cumulative Distribution Functions (CDFs) as one might expect in a probabilistic analysis, but are actually CBFs and CPFs for each output. As discussed earlier, probability theory can be viewed as a restrictive case for evidence theory, and therefore the two are compatible if viewed appropriately. This "band" seen for each output corresponds to the potential range over which the output can fall. Unlike in probability theory representations, one cannot make the statement that a given value is expected or that there is a given confidence that the analysis will produce a result above or below a given value. When interpreting results from a probabilistic study, the designer is able to make statements such as "there is a 95% probability that Y will be less than the maximum (safe) value", and therefore, if this is acceptable, the design can move on. If the same study were repeated with an evidence theory approach, a statement that exact is not possible; instead, the designer could make a statement such as "there is a possibility that Y will exceed its maximum (safe) value", and therefore, one should either change the design or select a different M&S environment with smaller "adder" functions. This would be a more conservative statement than one achieved with probability theory, but it is more appropriate based on the available information.

2.5 Summary

This chapter summarized a literature review conducted to identify a promising method for uncertainty quantification while addressing the FSP. Both reliability engineering and M&S perspectives on uncertainty were examined and compared, allowing the reader to see how the epistemic model uncertainty being addressed in this study compare to these commonly

used taxonomies. Specific aspects of the FSP prohibit the use of more mature probabilistic methods for uncertainty quantification. Other theoretical methods were surveyed and compared. The most promising method examined was evidence theory, a method initially proposed by Dempster in the 1960's and further developed by Shafer in the 1970's. Two issues with an evidence theory approach were identified. A solution to the first issue, the need for a clear definition for belief and plausibility (the two fundamental measures of evidence theory), was provided in this chapter. The second issue is the extreme computational cost associated with appropriately quantifying uncertainty with evidence theory. This issue will be discussed in the following Chapter. Once completed, this research will give users the ability to appropriately quantify all relevant types of uncertainty in their M&S environment, therefore allowing them to more effectively completed any uncertainty-based modeling tasks.

CHAPTER III

SURVEY OF RELEVANT UNCERTAINTY PROPAGATION LITERATURE

3.1 Introduction

In the previous chapter, evidence theory was selected to quantify epistemic model uncertainty. Because evidence theory could be thought of as a generalized form of probability theory, many of the sampling methods commonly used in probability theory can be leveraged with some modification. This is advantageous because probability propagation methods are relatively mature and the methods used today have been developed over the previous 70 years.

In this chapter, commonalities in probabilistic uncertainty modeling will be identified and the alternatives will be decomposed into various classes. The alternatives in each class will be discussed briefly from a historical perspective, in terms of their implementation, and their pros and cons. Following this a qualitative comparison will be performed and recommendations of most favorable methods will be provided based on the type of problem being considered. After identifying the most promising probabilistic methods, their modification for use as evidence theory propagation methods will be discussed.

3.2 Probabilistic Uncertainty Modeling

Probabilistic uncertainty propagation methods have two commonalities: a sampling subject and a sampling method. The two sampling subject alternatives available include the actual analysis and an approximation of the analysis. While the actual analysis is the most accurate sampling subject, it can be extremely expensive depending on the analysis being conducted. Response Surface Equations (RSEs) are closed form expressions which approximate the actual analysis or analyses and are commonly used in many engineering design applications. Second order polynomials and Artificial Neural Networks (ANNs) are commonly used RSE forms. Each RSE, regardless of its specific form, is created by strategically

sampling the input space for an analysis or analyses to gather as much information about their output space as possible while being penalized with as little computational expense as possible. These strategically taken samples compose a Design of Experiments (DoE), which in the context of computer simulations is also referred to as Design and Analysis of Computer Experiments (DACE) in some literature.[217] Once the Design of Experiments (DoE) samples have been taken, they are used to create the desired RSE.

When examining probabilistic sampling methods, one could say there are two classes of methods that can be used to conduct uncertainty propagation. The first class of methods repeatedly takes samples which after hundreds, thousands, or even millions of samples will form the desired input (and resulting output) Probability Density Function (PDF). This method is more accurate for engineering solutions than the alternative class of methods but it is extremely expensive to implement because of the large number of samples required to form the desired distributions. Methods in this class include Monte Carlo (MC) sampling, stratified MC sampling, Importance Sampling (IS), and Quasi Monte Carlo (QMC) sampling. The second class of methods takes a different approach by approximating the desired PDF by making some assumptions and then taking relatively few samples. This class of methods accepts that while the result may not be as accurate as additively building a distribution for some situations, these methods can often be completed in orders of magnitude fewer function calls (and equivalently time). This class of methods began with First Order Second Moment (FOSM) techniques, and includes Fast Probability Integration (FPI), Point Estimation Method (PEM), and methods developed in Multidisciplinary Optimization (MDO) applications.

3.3 Sampling Subjects

As discussed earlier, the first common element in a direct sampling technique is the subject being sampled. The simplest and most obvious sampling subject is the analysis itself. This is the simplest alternative to explain and is also the most widely accepted sampling subject. While the benefit to using the analysis code is that the results are as accurate as the analysis code, one major drawback is the clock time needed to complete the sampling, especially

if the analysis requires significant clock time to run a single case or if many inputs are being varied. The alternative to using the actual analysis tool is to use an approximation of the tool, commonly referred to as an RSE. In the literature, RSEs are also referred to as response surface approximations, meta-models, surrogate models, and transfer functions, depending on the community using the RSE. Several different forms of RSEs exist which can be used. These include both linear and nonlinear regressions.

Linear regression techniques, sometimes referred to as curve fits, are single or multivariate Taylor series expansions where the coefficients are obtained through a least squares, or similar, analysis. Commonly, these polynomial RSEs are linear, linear with interactions between the linear terms, or quadratic in nature, though some fields also calculate third, fourth, and other higher order terms. A common second order Taylor series expansion is given in Equation 8, where the first summation is the linear terms, the second is the quadratic terms, and the third is the linear interactions.[221] RSEs have been used in a variety of different ways, including optimization, design, and the acceleration of processes.[203, 212, 215]

Some benefits for using polynomial RSEs are that the equation can relatively inexpensively represent simple trends (linear terms), and that the relative magnitude of the coefficients allows the user to gain insight into the physical phenomena at hand. Large coefficients show the user which terms are more sensitive, and therefore important, than others, whereas smaller coefficients show the opposite, and in some instances they can be defaulted and removed from the analysis completely. Linear polynomial RSEs can be constructed using relatively few samples, which led to their popularity with DoEs in the agricultural community.

$$Y = \sum_{i=1}^{k} b_i x_i + \sum_{i=1}^{k} b_{ii} x_i^2 + \sum_{i=1}^{j-1} \sum_{j=i+1}^{k} b_{ij} x_i x_j \tag{8}$$

Nonlinear regressions are similar to linear regressions in that they can both serve as approximations for an analysis. They do significantly differ in their general form and creation method. In linear regression, the coefficients (b's in Equation 8) are constant. In nonlinear regression, these coefficients vary as a function of some other parameter or parameters.

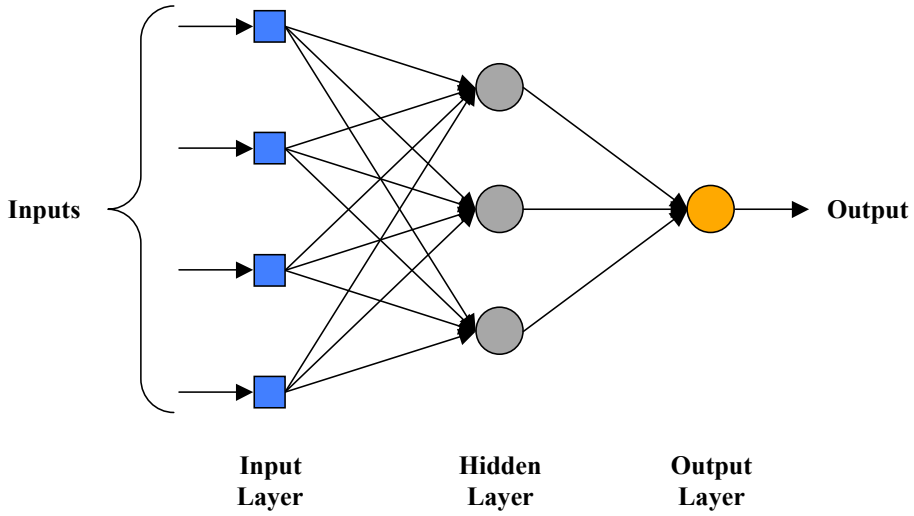


Figure 11: Architectural Illustration for MLPs RBFs, and SVMs.

One common type of nonlinear regression is an ANN. ANNs were conceived in 1943 when McCulloch and Pitts proposed a theory that linked a series of elementary units, which they called neurons because of the system's similarity to a biological brain, that could perform computational tasks.[158]

Since this 1943 paper, ANNs diverged into the variety of different systems which we have today. While there are a multitude of different types of ANNs, the three most commonly used types in MDO applications are Multilayer Perceptrons (MLPs)[69], Radial Basis Functions (RBFs)[216], and Support Vector Machines (SVMs)[31] Each of these is different, but they all have the same basic architecture, containing an input layer, a hidden layer or layers, and an output layer. This is illustrated in Figure 11. The input layer consists of nodes containing the inputs to the equation. A hidden layer(s) consists of several highly (or possibly completely) interconnected neurons (or nodes). Each of these edges is connected to the various neurons, which have a weight and bias. The third component is the output layer. This consists of the output or outputs which are being approximated with the neural net and the activation function which combines information from the hidden layer. This architecture is used because ANNs are intended to mimic the behavior of biological neural networks such as the brain.

MLPs are generally illustrated in Figure 11, but this could be thought of as a limiting

case. MLPs do consist of an input layer, a hidden layer, and an output layer, but there may be multiple hidden layers.[95] MLPs with multiple layers are commonly governed using feedback linkages whereas single hidden layer MLPs are governed with feedforward linkages. Single layer, feedforward MLPs may have more edges and nodes than a multi-layer feedback MLP, but the universal approximation theorem states that a single-layer feedfoward MLPs is able to approximate any function, given a sufficient amount of training data.[95] Training methods for MLPs are stochastic, meaning these weights and biases in the hidden layer or layers are stochastically determined to effectively "train" the highly interconnected MLP to approximate a desired function using a variety of different training algorithms. The number of hidden layers and neurons can be iteratively determined by beginning with a small number of neurons and one hidden node and increasing the complexity as needed to better approximate the desired function. This is done to avoid over-fitting, which is a common issue with ANNs. MLP output layer activation functions have been modified to approximate both classification and continuous problems. Because of the stochastic training process used with MLPs, more samples are generally needed to train an MLP than to develop a second order polynomial linear regression when there are fewer than approximately 10 dimensions, or input variables, being examined. One benefit to using MLPs rather than polynomial RSEs is that polynomial RSEs are unable to represent highly nonlinear behavior. Guidelines state that the number of points required is of the same order of magnitude as the ratio of the complexity of an MLP and the acceptable error of the model.[95]

RBFs, another common type of ANN, are also a common form of nonlinear multivariate regression used in MDO applications. Interestingly, RBFs were actually developed as a method for interpolation rather than regression, and then were later used for nonlinear regression.[135] The basic architecture for an RBF is provided in Figure 11. RBFs only have one hidden layer, much like a feedforward MLP. RBFs, though similar to MLPs in architecture, are different in their training methods and in their representations of the space as modeled with activation functions in the output layer. Because RBFs are intrinsically an interpolation algorithm, they model the global behavior of a system through many local

behaviors using a collection of basis functions as their name implies. The specific basis function selected is dependent on the problem at hand and is most easily selected through a trial-and-error process. MLPs, on the other hand, primarily model the global behavior and have a much more approximate, and in some cases noisy, representation of the local behavior. This difference can be seen in their training methods; MLPs can be said to recursively and stochastically train the network to approximate a function whereas RBFs can be said to curve fit through a function.[95] Also, because less information is required to perform a curve fit than to train a model, RBFs generally require significantly fewer points for fitting than either a second order polynomial RSE or an MLP.[147] Like MLPs, RBFs are able to approximate both continuous and classification functions, whereas polynomial RSEs are only able to approximate continuous functions. While RBFs require fewer points to train than MLPs, they can also experience over-fitting issues if inappropriately trained.

SVMs are a third common type of ANN used to solve both classification and regression problems.[95] SVMs are also referred to as Support Vector Regressions (SVRs) in some literature.[31] One should note that while the name is different, the machine in an SVM is performing a regression so they are in essence the same. SVMs, like feedforward MLPs and RBFs, have only a single hidden layer as shown in Figure 11. SVMs take a slightly different training approach than either MLPs or RBFs. SVMs identify an optimal hyperplane in the training data. For a classification problem, this would be the hyperplane separating one choice from another. The optimal hyperplane is similarly located for a regression problem. When calculating the optimal hyperplane, a series of points closest to the optimal hyperplane is identified; its distance from the optimal hyperplane could be thought of as a "support vector". In essence, SVMs are a machine, or algorithm, composed of these support vectors which are a subset of the training data. The complexity in training an SVM is in identifying the optimal hyperplane, which is commonly accomplished with Nondeterministic Polynomial (NP)-hard constrained optimization techniques. The concept of optimal hyperplanes was pioneered by Vapnik and Chervonenkis in the 1960s.[213] The SVMs, though distinct from both MLPs and RBFs, could be thought of as a more general form of ANN when used for either regression or classification problems. In the case of both

MLPs and RBFs, additional information is assumed about its form (RBF) or training procedure (MLP) that somehow simplifies the problem. If these assumptions were removed, one would be left with an SVM. Therefore, one could say that the only real difference between an MLP, RBF, or SVM is the activation function and training procedure adopted.[95, 117]

Whether one chooses linear or nonlinear RSEs as a surrogate for the actual analysis, an additional step must be taken to create the RSE. To do this, a DoE is commonly used. DoEs were developed in the agricultural community in the early 1900's as a method for enhancing desirable properties in crops. The techniques were accepted in the statistical community and have become an integral part of industrial organizations' design processes in the last 20 years.[159] Because DoEs were not initially developed for computer experiments, a distinction between non- computer-based DoEs and computer-based DoEs, or DACEs, is sometimes made in the literature, though it will not be made in this study.[217] The purpose behind a DoE is to take as few samples as possible while capturing the maximum amount of information from the experiments. One should note that the choice of RSE dictates the type of DoE which should be selected. Information on this topic can be found in a variety of sources, including Myers & Montgomery[159] and Wu & Hamada[221]. Once the analysis has been used to run a DoE, the regression of choice can be conducted and the RSE can be obtained. This RSE can then be sampled from in lieu of the analysis code. The benefit to using an RSE rather than an analysis code exists if there are a large number of samples that must be taken in the sampling method, and therefore the time required to take these samples outweighs the time required to run a DoE and make the regression.

A final note on sampling subjects is that the subject selected should be determined based on the sampling method selected. Additive sampling methods require a significant amount of time to complete, so if RSEs of an acceptable accuracy can be created, they should be used instead of actual analysis code, given that the number of samples to additively form the desired PDF is less than that required to make an RSE approximating the analysis. Distribution approximation methods were developed to reduce the number of executions required through a series of assumptions. Because implementing the approximation method would require fewer function calls than developing a DoE, it only makes sense to select the

actual analysis code as the sampling subject instead of an RSE when using a distribution approximation method. Conversely, if a user were taking more samples with a distribution approximation method than were required to run a DoE, the user should instead use an additive sampling method, in essence removing the assumptions involved in the distribution approximation method.

3.4 Additive Distribution Sampling Techniques

In order to propagate uncertainty, one represents a non-deterministic phenomenon, and should therefore turn to non-deterministic methods of representation. An issue exists because analysis and design tools are generally deterministic in nature, but these tools are being used to mimic the non-deterministic phenomena. One method for representing this nondeterminism is simply to take hundreds, thousands, or millions of samples and assess them in the deterministic tool or an RSE of the tool. By strategically selecting these samples, the desired frequency distribution, or PDF, can be constructed which represents the non-deterministic phenomena of interest. The first method used to conduct this sampling was MC sampling. Later other methods, including Importance Sampling (IS) and stratified MC sampling, were developed which maintained many of the favorable aspects of MC sampling while reducing the number of samples which must be taken. An alternative method to MC sampling which shows promise is QMC sampling. Each of these methods will be outlined and pros and cons for each will be discussed.

3.4.1 Monte Carlo Sampling and its Derivatives

MC sampling is a stochastic sampling method developed by Los Alamos National Laboratory in the 1940's in support of the Manhattan Project.[100] This involves using a pseudo random number generator to place samples in a single bin for each dimension. As the number of samples increase, the frequency distribution of the bin will approach a uniform distribution. This method was extremely powerful when first developed. Prior to its advent, complex integrals were generally evaluated numerically. These expressions were complex and extremely time consuming to solve, meaning MC sampling was a significant improvement over the then-current state of the art. Also, because of the limitation of requiring a

closed form expression for non-deterministic parameters, these equations were difficult to evaluate. MC sampling not only allowed reliability engineering problems to be solved more easily, but it also allowed problems in statistical mechanics, fluid mechanics, and quantum mechanics to be evaluated more easily.[152] While this method was an extremely powerful and revolutionary way of examining complex problems, it was extremely computationally expensive. As stated earlier, as the number of samples increases, the distribution approaches a uniform distribution. Depending on the application at hand, the need for the sampling distribution to resemble a uniform distribution varied. In order to determine the sample size, one should consider how the first four moments (mean, variance, skewness, and kurtosis) of the created histogram compare to the true analytic distribution's moments and also consider the samples space filling properties. If a relatively low level of certainty in the solution is needed, meaning a moderate difference between the sample's distributional moments and the true moments is acceptable, then perhaps on the order of hundreds of samples are needed. This was the case with many MC implementations prior to the 1980's. However, if a high level of certainty in the solution is needed, as is the case in many current reliability engineering studies, many, many more samples must be collected. In modern reliability applications, it is not uncommon for millions of samples to be taken to ensure the safety of a critical product.

This extreme computational expense required for a high certainty in MC sampling results led to an adaptation of the MC sampling method, IS, whose development also dates back to the 1940's nuclear physics calculations.[98] Depending on the community, IS is also referred to as fast or quick simulation.[98, 196] IS was developed with the goal of approximating the results of an MC sampling in a more cost effective manner. If the same number of samples for IS and MC sampling were taken, there would be a Variance Reduction (VR) from the sampling and the true analytic solution, or equivalently if the same variance were obtained, it could be achieved with fewer samples. This approach realizes that in most problems, there is a relatively small "interesting" region of the input space and that much of the input space is "uninteresting".[111, 132] To take advantage of this realization, a non-uniform (biased) MC sample is created where a relatively high frequency of psuedo-randomly generated

samples are in the "interesting" region while comparatively fewer samples are taken in the "uninteresting" region, thereby gaining the information of a much larger MC sampling in the "interesting" region while only running a fraction of the samples. Once the samples are analyzed, a weighting is applied to the samples to produce an unbiased estimator. The equation for this weighting is given in Equation 9.

$$E = \int_A \frac{f(x)}{h(x)} h(x) dx \tag{9}$$

Where:

1. E is the unbiased estimator
2. A is the area being integrated over (input space)
3. $\frac{f(x)}{h(x)}$ is the weighting applied to the biased sample
4. $h(x)$ is the bias sample, or PDF which was evaluated

The problem with implementing IS is that the "interesting" region must be known a priori. A variable fidelity approach is commonly used to address this. First order methods such as FOSM methods were commonly used up through the 1980's and 1990's to generally identify failure regions (and will be discussed in a subsequent section of this chapter), enabling IS to then be employed.[70, 149, 150] If this "interesting" region is appropriately identified, the answer will be of a similar accuracy to a much larger MC sampling, though if inappropriately selected, it will result in a much poorer result.

The question remains in IS as to how these biased, or non-uniform distributions, should be created. There are two basic methods used to create non-uniform distributions: rejection sampling and inverse transformation.[53] The first and simplest is a rejection sampling approach. Here a uniform distribution is created and samples are strategically removed, or rejected, from the uniform distribution to produce a non-uniform distribution of interest. While applicable for purely MC samples, a rejection sampling approach would negate any favorable features a given uniform sample may posses. The second alternative is to create

non-uniform distributions through inverse transformation. This operates by beginning with the property provided in Equation 10:

$$F(X) = U \tag{10}$$

Where:

1. U is a uniform random sample
2. X is the desired non-uniform random sample
3. F is the Cumulative Distribution Function (CDF) of X

Equation 10 can be rearranged to the following:

$$F^{-1}(U) = X \tag{11}$$

Using this more useful form in Equation 11, the inverse CDF of F can be applied to a uniform sample, producing the desired non-uniform sample. While the expression for F^{-1} must be obtained through numerical methods for arbitrary distributions, many statistical and mathematical software packages have expressions for common, non-infinite distributions (triangular, beta, etc). Because of its ability to maintain any favorable properties of the uniform sample, this method is favored over a rejection sampling method.

A second common VR technique applied to MC sampling is stratified MC sampling. Stratified MC sampling is a VR technique that uses a similar approach as MC sampling in that a pseudo random number generator is used to populate a frequency distribution forming a uniform distribution. The adaptation is that rather than having a single bin, as was the case with the original MC implementation, a series of bins, or strata, are distributed in each dimension.[101]

The stratified MC sampling benefit is that in a given dimension, the true distribution's moments can be approximately reached in orders of magnitude fewer points than with MC sampling. However, stratification of the samples in a given dimension does not necessarily improve the space filling properties or correlation of the samples in other dimensions. It is

unknown how points in the tails of a stratified MC sampling will relate to points in other input spaces. Therefore, there exists a distinct possibility that the bounds of a given input space will not be adequately covered. This problem was identified and addressed in the development of Latin Hypercube Sampling (LHS). This sampling method was developed in the 1970's at Sandia National Laboratory, while working with the U.S. Nuclear Regulatory Commission (NRC), as a response to criticism the NRC had received with respect to its probabilistic risk assessment's inadequacy to characterize uncertainty propagation when modeling nuclear systems.[101]

LHS uses strata similar to stratified MC analysis. However, LHS requires that there be an equal number of strata in each dimension, that these strata are uniformly sized, and that only one point is placed in each stratum. Equivalently, the number of strata and the number of samples are the same. If done in only two dimensions, this technique approximates an orthogonal Latin Square.[112]

Due to these favorable properties, one could say LHS is the most popular stratified MC sampling method used today. Although stochastic in nature, the orthogonality of LHS can be maximized through iteration, ensuring a better coverage of the input space. One unique feature about LHS, where it differs from MC sampling, is that while MC and stratified MC sampling have pre-assigned distributions, LHS is essentially a "blank" distribution. After running an LHS the samples can be weighted in a post- processing exercise to represent results from any arbitrary distribution with minimal bias.[112]

Each of these three methods, MC sampling, IS, and LHS, are illustrated by examining two dimensions with 40 samples for each method in Figure 12. In Figure 12 the red dashed circle represents the "interesting" region for IS and the individual strata for LHS are shown with black dotted lines. In this figure, the reader can see large holes in the MC sampling on the far left. Similarly, the IS pane in the center has a relatively thorough sampling in the "interesting" region as desired. In the right pane, the reader can see how LHS far outperforms MC sampling with respect to uniformly covering the input space for these two variables.

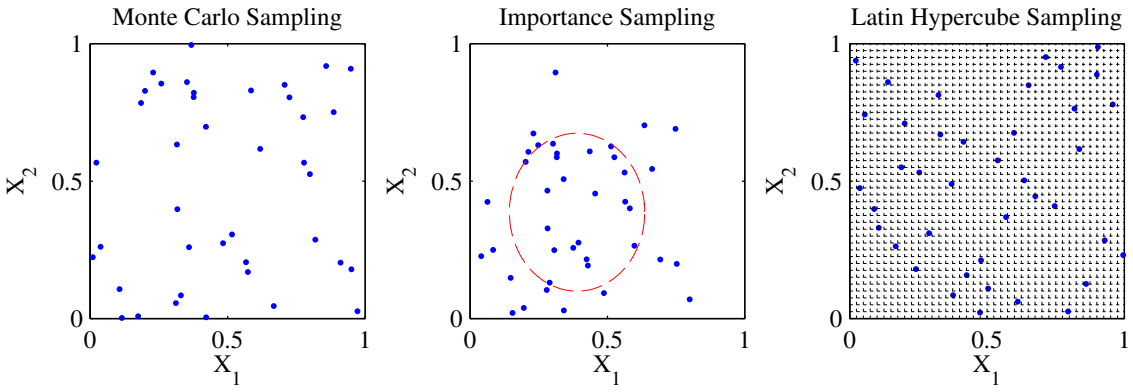


Figure 12: Comparison of MC, LHS, and IS Methods.

3.4.2 Quasi Monte Carlo Sampling

All other MC sampling methods discussed up until this point have taken "random" samples to describe a distribution. This is because the basis for a MC sample was with the idea that truly random numbers could be used. However, this is not actually the case. These random samples are not actually random, but are in fact *pseudo* random and are deterministic in nature. While in many ways these pseudo random sequences pass statistical randomness tests, they are not truly random. Lists of pseudo random numbers were published and used before computers began to use essentially the same list of "random" numbers and choose a point to start in the list (seed) as is done today.[207]

While these pseudo random sequences pass statistical randomness tests, they also possess undesirable traits. Specifically, the error or discrepancy between the result produced with pseudo random sequences and what theoretically occurs for the distributions are both nondeterministic due to the changing seed and require a significant number of samples to converge upon a solution.[163]. One alternative to using these pseudo random sequences of numbers is to use a sequence of numbers with points that are strategically placed, rather than pseudo randomly placed. This is the objective of QMC sampling sequences, which are also referred to as "low discrepancy sequences" because of their lower discrepancy relative to classic MC sampling.[181] Several different types of QMC sequences exist. The more popular sequences derive from two common sequences, one involving prime number bases for a radix sequence which essentially "steps" through the input space to uniformly fill it,

and a second that starts with points spread equally in a unit volume, or (t,s) net, and then replicates this net to fill the space.

The prime number base sequence was first proposed by Van der Corput, a Dutch mathematician, in 1935.[164] While it might seem interesting that this was published before computers were used for random sampling, one should note that distribution sampling is not limited to computer-based MC approaches but actually goes back as far as "student's" work by William Sealy Gosset. Gossett published his statistical work from wort batch fermentation quality monitoring at Guinness under a pen name ("student") from 1908 through 1925.[174, 207] Van der Corput simply developed an alternative approach to pseudo random sequences which works using a prime number radix in one dimension. This work inspired Halton and Hammersley to develop their own variants on the Van der Corput sequence. Hammersley first published his variant on Van der Corput's sequence as an alternative to using MC for numerical integration because his sequence had a lower discrepancy.[89] Soon thereafter Halton published a slightly different variant of Van der Corput's sequence.[88] Both Hammersley's and Halton's sequences expand the 1-dimensional Van der Corput sequence to an n-dimensional sequence. They do so by assigning a different prime number as the base for these new dimensions. For example, the first dimension of either sequence has a radix base of 2, the second dimension has a radix base of 3, the third a radix base of 5, the fourth a radix base of 7, and so on.

Halton's sequence, though similar to Hammersley's, was different in the way radix normalization was conducted. Because all sampling is done in a unit volume, Hammersley normalized the results based on the largest resulting base which is determined by the number of samples taken. Halton took a slightly different normalization approach, normalizing each radix by its base rather than by the largest base. This difference in normalization allows Halton's sequence to be an infinite one whereas Hammersley's sequence is not.

One problem identified by many in the 1970's was that the Halton and Hammersley sequences did not possess low discrepancy in higher dimensions even though their originators had theoretically shown that they should. This is due to the nature of Van der Corput's sequence. As stated earlier, the radix essentially "steps" through a given dimension with

its prime base. When looking at higher dimensions (and therefore higher prime numbers) there are more samples required to complete the radix. Therefore many sequences will show bands of points with large gaps in between the bands. This is caused by correlation occurring between the different prime bases, which in turn will lead to a poor coverage in 2-D cross sections of the sample space. This is illustrated in left-most pane in Figure 13. A similar issue occurs in the Hammersley sampling sequence

This problem is significant, but not a fundamental failure of the method. The sampling will be completely uniform if the radix is completed, but this will require the number of samples to be the product of the prime bases for that 2-D slice of the unit hypercube. Take, for illustration purposes, the "banding" which occurs between the 39^{th} and 40^{th} dimensions of a Halton sampling sequence, as shown in Figure 13. The number of samples required for the 39^{th} and 40^{th} dimensions sequence to be complete is the product of the 40^{th} and 41^{st} prime numbers, or 167 multiplied by 173, or $28,891$ samples, which means while it is not a fundamental failure, the solution is infeasible. This infeasibility was addressed by Braaten & Weller when they published their 1979 work on "scrambled" Halton sequences, later dubbed a "leaped" Halton sequence by others. This sequence involves leaping through the radix, therefore finishing the radix more quickly, and eliminating, or at least reducing, the banding in higher dimensions. Therefore, rather than moving through a sequence from $i = 1, 2, 3, ...n$, one would move through it as $i = 1k, 2k, 3k, ..., n$ where k is a leaping number. Braaten & Weller conducted several empirical studies and identified several promising radix leaping numbers, all of which were primes. A later development by Kocis & Whiten[124] involved reversing the radix, which essentially "clumped" points but did serve to eliminate the poor banding property in higher dimensions. This process was referred to as Halton RR2 sequence. In 1999, Robinson & Atcitty published a paper comparing quasi and pseudo MC sampling methods. Their work outlines the weakness of the original Halton sequence described here and also implemented the Leaped Halton and RR2 Halton sequences. These methods were compared against LHS. It is interesting to note that the authors found the Leaped Halton sequence to outperform LHS for a variety of problems, especially those that are highly non-linear. The authors also pointed out that some instances (though

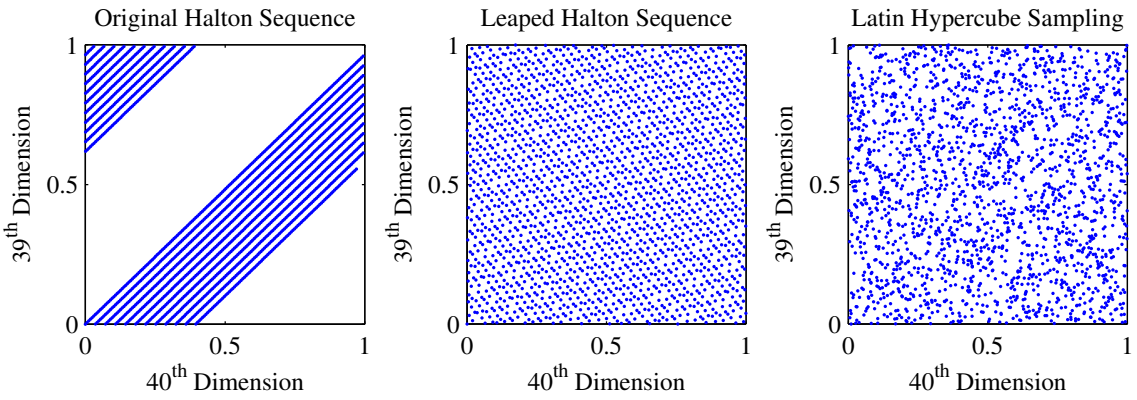


Figure 13: Original and Leaped Halton Sequences Compared to an LHS

interestingly not elaborated upon their work) exist where QMC methods fail to adequately predict while LHS succeeds. A comparison of a 2000 sample original Halton sequence and leaped Halton sequence (leaping base = 409) can be found compared to an LHS for the 39^{th} and 40^{th} dimensions in Figure 13.

The second approach, using a (t,s) net, will not be explored for the purpose of this research. These sequences exhibit similar "banding" in highly dimensional problems as found in Van der Corput sequences. Work by Kocis & Whiten showed that Sobol and Faure sequences, which are derived from (t,s) nets, have a higher discrepancy than Van der Corput-derived sequences when leaping is applied. [124] For more information on both Van der Corput- derived and (t,s) net-derived QMC sequences, please consult a thorough literature survey by Niederreiter.[164]

The comparison between various QMC sequences and LHS, done by Robinson & Atcitty, showed that a leaped Halton QMC sequence is preferable to LHS. This author has found the same to be true, but with some reservation. Leaped Halton sequences were derived in an effort to better distribute samples in higher dimensions. This is successful, but as shown in the Braaten & Weller study, it is not completely successful, and is highly dependent on the leaping (or scrambling) prime base selected. Benefits to using a leaping Halton QMC sequence are that it can be very quickly generated, it is deterministic, and given the appropriate leaping base it may have better space filling properties than an LHS. Disadvantages to using a leaped Halton QMC sequence are that the appropriate leaping base must be

identified, and that this leaping base is dependent both on the number of dimensions being examined and the number of samples in the sequence. While authors including Robinson & Atcitty and Braaten & Weller do examine a 2-D scatterplot to ensure a lack of banding, this is a very limiting approach. As one changes the leaping base, the "banding" in one dimension may be removed, but there is a very real possibility that it could instead appear in another dimension. For this reason, one shouldn't simply examine one 2-D scatterplot, but instead examine all 2-D scatterplots, and this has not been done in the literature.

3.5 Distribution Approximation Techniques

Because of the extreme cost with running hundreds, thousands, or millions of points, various approximation methods were developed. These methods, though not as accurate for certain types of problems, offer a significant time savings over additive sampling methods. Several methods will be discussed, including the FOSM methods, FPI, PEM, and other methods leveraged from MDO techniques. These methods require differing amounts of time to implement, and correspondingly produce results of differing levels of accuracy. Each method should be used with care because the results are only as valid for a given problem as the assumptions in the method. Each of these methods will be outlined in roughly the order they were developed.

3.5.1 First Order Second Moment Method

FOSM methods are a class of reliability methods developed in the civil engineering reliability field. Dating back prior to the 1940's civil engineers were examining the non-deterministic properties of materials. This was done primarily using probability theory to quantify uncertainty in a system and propagate it through to metrics of interest. One example of how this was examined is as follows. An object with a deterministic resistance, R, would be subjected to a nondeterministic stress, S. The safety margin, M, is defined as $R(A) - S$.[42]

The "fail" set, designs which would fail because $M \leq 0$, and the "safe" set, designs which would not fail because $M > 0$, could be identified with this analysis and the probability of failure could be quantified. Similarly, the "limit state" or "failure function" is defined to be all instances were $R(A) = S$.[85, 223] In this probabilistic approach, the probability of

failure can be solved directly through integration, as shown in Equation 12.

$$P_f = \int_{\Omega} f(x)dx \tag{12}$$

Where:

- Ω is the failure set
- $f(\mathrm{x})$ is a probability density function of one or many variables

In many simple problems evaluating this integral can be done easily. However as the complexity of the problem and number of variables increase, the expression becomes increasingly difficult to evaluate. As an alternative to evaluating Equation 12, many reliability engineers turned to MC sampling techniques which were discussed in a previous section. While simpler than directly solving the integral, MC sampling of an analysis code can be time consuming with today's computers and was significantly more costly in the 1950's when engineers were investigating these methods. A second alternative to both evaluating Equation 12 and MC sampling was an FOSM method. The method is "first order" because it uses a linear approximation of $f(x)$ that uses derivative information about $f(x)$ and is a "second moment" method because it is approximating the second moment, or variance, of a PDF. To perform the linear approximation in FOSM, a linear Taylor Series expansion was commonly used, and because of the expansions widespread popularity, FOSM and Taylor Series expansion are used interchangeably in some texts.[220] From this information, the safety index, β, could be calculated. The safety index is given in Equation 13[55]

$$\beta = \frac{E[M]}{D[M]} \tag{13}$$

Where:

- E is the expected value or mean for M
- D is the standard deviation for M

The downfall to this FOSM approach is its lack of invariance[1]. While the formulation provided above ($M = R(A) - S$) is an adequate, appropriate way to quantify the safety margin, another valid expression is $M = R - S/A$. Through this simple transformation the same system will produce two different safety factors. For this reason, the method was difficult to implement, and the specifics of its implementation had to be carefully documented.

More recently, Second Order Second Moment (SOSM) methods have been proposed.[43] While they do increase the failure probability prediction, they also have a significantly increased time for implementation because of the additional function calls needed for a second order approximation. Because of the rise in popularity of other methods including PEM and various MC sampling methods, SOSM methods are not widely used today.

It should be noted that while this example was given in one dimension, the method was formulated as a multivariate one. It is intrinsically assumed that all inputs are uncorrelated and independent, though this is not always the case in reality. To correct for possible correlation and dependence in the input space, it was common practice to use the transformation published by Rosenblatt in 1952.[182]

3.5.2 Fast Probability Integration

In 1974, Hasofer and Lind published a slightly different formulation for calculating β relative to FOSM as it was practiced at the time. They stated β should be the ratio between the standard deviation and the distance to the failure point where $R = S$.[94]. While their method was limited to only normal distributions, it was invariant and addressed the previous issue with FOSM. This method was generally seen as an improvement, but unfortunately only worked for normal distributions and disregarded all other information about a PDF that might have been available. This new Hasofer and Lind safety index became known as β_{HL} named after the authors who proposed it. Later, in 1978, Rackwtiz and Fiessler published a paper outlining a method which approximated the tails of a distribution by transforming a distribution into an "equivalent" normal distribution and then calculating

[1] In a mathematical content, invariance means the result is unchanged even when the equations undergo a transformation

β_{HL}. Though similar to FOSM, it was different because the method calculated β_{HL}. Because this new method could more quickly find the probability of failure than either direct integration or MC sampling, it was called "fast probability integration" by others in the reliability engineering field.[55] Soon thereafter Chen and Lind proposed a three parameter method for approximating a normal distribution tail like Rackwitz-Fiessler (R-F) had, and felt that their additional parameter (R-F uses 2 parameters) could more accurately predict the failure probability.[29] Results shown by Wu show that while the Chen-Lind (C-L) three parameter method does have a better prediction performance than the R-F method for some problems, it does not consistently predict a better result.[223] Ditlevesen also developed a method for normal tail approximation using a similar process in the early 1980's.[56]

In 1984, Wu wrote his dissertation on an approximation method which combined and improved the R-F and C-L methods and called it a "fast probability integration" method as did many others in the reliability engineering field at the time.[223, 224] Wu went on to work at the Southwest Research Institute (SwRI) where they incorporated his version of FPI into Numerical Evaluation of Stochastic Structures Under Stress (NESSUS), a tool they were developing under contract for NASA. Here the method gained further popularity and has been used by a variety of organizations to perform first order reliability and other probabilistic analysis.[145, 156]

It should be noted that some authors consider FPI and other distribution approximation methods to be a part of FOSM because this was the motivation behind developing FOSM. For the purposes of this paper this distinction will be made because FPI and certain other distribution approximation methods differ from FOSM as formulated in the 1960's, and are still used by some in the reliability field.

3.5.3 Point Estimation Method

PEM is a method initially developed by Rosenblueth in 1974. Rosenblueth pioneered a method where the first and second moments of a probability distribution can be approximated with 2 points for a distribution in one dimension. In multiple dimensions, the number of samples grows by 2^n, where n is the number of uncorrelated input variables. This occurs

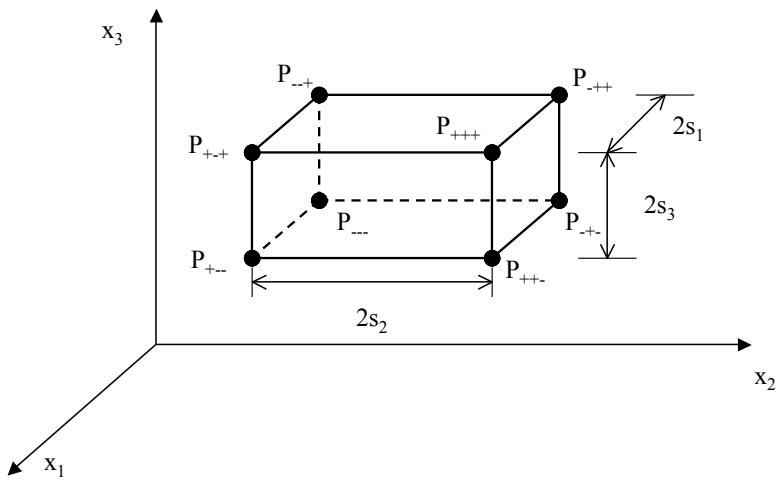


Figure 14: Rosenblueth's PEM Sampling Procedure in Multiple Dimensions.[184]

because Rosenblueth's version of PEM essentially samples the corners of the hypercube. This is depicted in Rosenblueth's 1975 article and in Figure 14.

While there is an extreme cost of sampling for highly dimensional problems, this method has worked well for low dimensional geotechnical and civil engineering reliability problems and it served as a foundation for later developments. Lind showed that for uncorrelated input variables, Rosenblueth's original formulation could be re- examined and the number of samples could be reduced from 2^n to $2n$.[136] This method essentially calculates the face-centered points of the hypercube rather than the corners of the hypercube. Harr showed that for nonlinear outputs, a 3-point PEM could easily be used, but the number of calculations per input variable was more expensive than Rosenblueth's original formulation, requiring 3^n samples.[92] More recently, a hybrid method was developed by Wang et. al. and published as an adaptive PEM where the user would first determine the number of points necessary for each input variable and then appropriately sample for high dimensional problems.[217] PEM has been used in a variety of reliability and civil engineering studies for over 25 years.[92, 108, 133, 136, 183, 233]

Interestingly, a similar method was later independently developed in 1983, based more in mathematics. Later it was shown that PEM is a Gaussian quadrature[2].[30, 155, 231]

[2]A quadrature from a numerical integration perspective in two dimensions is simply the calculation of the area under a curve, or in this instance, a PDF.

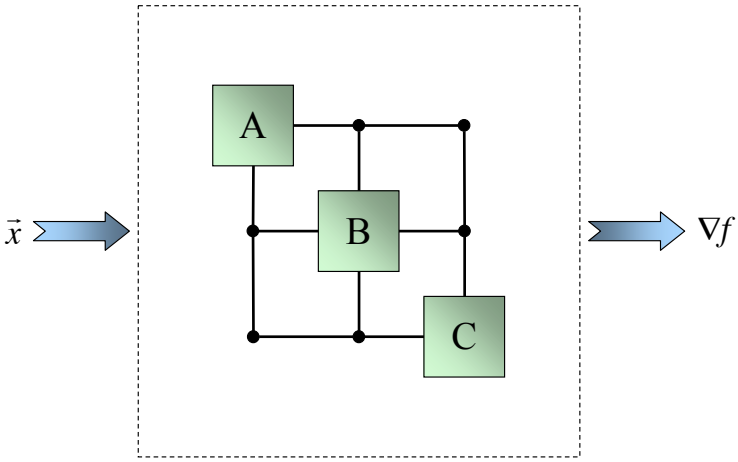


Figure 15: Fully Coupled DSM.

This is not as widely used as Rosenblueth's PEM.

One should use caution when using PEM; while the method does produce a result with significantly fewer function calls than additive sampling methods, it intrinsically assumes the response is monotonic with respect to the input variables. PEM has been shown to outperform FPI for noisy responses, because PEM does not take derivatives, simply points, whereas FPI uses derivative information to perform approximations. For non-monotonic responses, PEM has also been documented to give a poor predictive power.[216]

3.5.4 Multidisciplinary Optimization Techniques Modified for Uncertainty Propagation

While all methods discussed up until this point have been produced by the civil engineering or reliability engineering fields, there are also MDO techniques that have been leveraged to perform uncertainty propagation. Specifically, the sensitivity-based Global Sensitivity Equation (GSE), developed by Sobieszczanski-Sobeiski[197] has been used to propagate uncertainty by some. These methods stem from Sobieszczanski-Sobeiski's second form of the GSE (his first form involves residuals) and he refers to the more widely used form as GSE2. For completeness, an abridged derivation of Sobieszczanski-Sobeiski's GSE2 equation, as applied to uncertainty propagation, was performed for the fully coupled Design Structure Matrix (DSM) containing three Contributing Analysiss (CAs) as depicted in Figure 15.

For this fully coupled DSM, one can write an equation for the derivative of each CA's

output using the chain rule as shown in Equation 14.

$$\begin{aligned} dY_A &= \frac{\partial A}{\partial x}dx + \frac{\partial A}{\partial B}dB + \frac{\partial A}{\partial C}dC \\ dY_B &= \frac{\partial B}{\partial A}dA + \frac{\partial B}{\partial x}dx + \frac{\partial B}{\partial C}dC \\ dY_C &= \frac{\partial C}{\partial A}dA + \frac{\partial C}{\partial B}dB + \frac{\partial C}{\partial x}dx \end{aligned} \tag{14}$$

Rearranging this and rewriting it in matrix form produces Equation 15

$$\begin{bmatrix} I & -\frac{\partial A}{\partial B} & -\frac{\partial A}{\partial B} \\ -\frac{\partial B}{\partial A} & I & -\frac{\partial B}{\partial C} \\ -\frac{\partial C}{\partial A} & -\frac{\partial C}{\partial B} & I \end{bmatrix} \begin{bmatrix} \frac{dA}{dx} \\ \frac{dB}{dx} \\ \frac{dC}{dx} \end{bmatrix} = \begin{bmatrix} \frac{\partial A}{\partial x} \\ \frac{\partial B}{\partial x} \\ \frac{\partial C}{\partial x} \end{bmatrix} \tag{15}$$

The leftmost matrix is referred to as the Local Sensitivity Matrix (LSM). The right most vector is called the Sensitivity Derivative Vector (SDV), and the center vector of total derivatives is called the Local Sensitivity Vector (LSV). This is a powerful analysis for decoupling complex systems. By doing this, the user can see how a given analysis varies with respect to the input $\overrightarrow{x}$ in the LSV. Similarly, looking at the off-diagonal terms in the LSM, the user can see the coupling between the different CAs. To solve for the LSM, which is useful in optimization applications, Equation 15 should be rearranged to the form given in Equation 16.

$$[LSV] = [LSM]^{-1}[SDV] \tag{16}$$

This basic GSE2 was leveraged by Gu et. al. in 2000, where she re-derived Sobieszczanski-Sobeiski's GSE2 with a slight modification.GSE2 could be thought of as a sensitivity analysis, where given some change (or uncertainty) in an input parameter, one could quantify the change (or uncertainty) in an output parameter. Gu used this, but also accounted for the possibility in a bias in each CA. This form was intended to account for both aleatory and epistemic uncertainty, though Gu used a slightly different uncertainty nomenclature. This modified GSE2 was then used to propagate worst case uncertainty bounds, or equivalently perform interval analysis. As an alternative to an additive sampling approach with PDFs,

Gu showed that by propagating the upper and lower bounds of the input and bias uncertainty parameters' PDF, one could identify the bounds for the uncertainty in the output parameters.[84]

McDonald showed in 2006 that GSE2 can be used to propagate normal distributions, where the standard deviation is used to quantify the normal distribution as a single value.[146]

The difference between Gu's interval analysis and McDonald's method is in how the inputs were defined and what was done with the results in post-processing. McDonald propagated the standard deviations of normal distributions and accounted for covariance between the various inputs. This allowed him to reconstruct normal distributions for the outputs after propagation. Gu's interval analysis was effectively propagating a box with bounds representing a worst-case scenario. For a more detailed derivation of the GSE2, the reader should consult Sobieszczanski-Sobeiski's conference publication, Gu's article, or McDonald's dissertation. For reference, Gu's final form of the modified GSE2 is provided in Equation 17 and McDonald's final form of the GSE2 is given in Equation 18.

$$\begin{bmatrix} \Delta A \\ \Delta B \\ \Delta C \end{bmatrix} = [|LSV|]\,|\Delta x|] + [|LSM|]^{-1} \begin{bmatrix} |\delta_A| \\ |\delta_B| \\ |\delta_C| \end{bmatrix} \tag{17}$$

Where:

- ΔA is the bound in the responses for CA A
- Δx is the bound on the input variable(s) for the DSM associated with the inherent randomness (aleatory uncertainty)
- $\delta_A = \delta_A(x, B, C)$, is the potential bias (epistemic uncertainty) in CA A from inputs x, B, and C

$$\sigma_A^2 \approx ([LSM]^{-1})^2 \left[\varsigma_A^2 + [|SDV|]^2\,\sigma_x\right] \tag{18}$$

Where:

- ς_A is the covariance in the response of A with respect to the other CAs
- σ_x is the standard deviation of the uncertainty distribution on x being propagated through the DSM
- σ_A is the standard deviation of the uncertainty in A caused by σ_x being propagated through the DSM

One key assumption in GSE-derivative methods is in how they represent the results. Looking back to the GSE2 derivation, one can see it is a linearization of a complex, coupled system. While this is accurate for linear systems and for small perturbations of nonlinear systems, this limitation should be realized to ensure accurate uncertainty propagation. Uncertainty propagation with a GSE-derivative method can be performed more quickly than with FPI or PEM, but one should realize that not only does a GSE-derivative method assume the system is monotonic (meaning the derivative of each input with respect to the output does not change signs), and that the system is linear (meaning the derivatives are constant).

3.6 Summary of Uncertainty Propagation Sampling Methods

While a variety of different sampling methods have been identified, each with their own strengths and weaknesses, a comparison of these methods must be performed so that the most promising methods can be selected. To perform this comparison a series of Figures of Merit were identified. The various methods will be qualitatively assessed for each FoM and then a selection can be made. The four Figure of Merits (FoMs) identified by the author were:

1. Amount of time needed to use the method.
2. Ability to handle non-monotonic models.
3. Ability to handle higher order problems.
4. Scalability of the method (both time required and quality of result).

The amount of time required to run the method is the amount of time it would take for a user to run the method, excluding setting up and debugging. This was considered as an FoM because one important thing that differentiates the additive sampling methods from the distribution approximation methods is the amount of time required. The second FoM, the method's ability to handle non-monotonic models, indicates how well the method can handle situations where $\frac{\partial x_i}{\partial Y_i}$ changes sign. One example of this would be a sine wave when examining more than one period. This was considered important because many complex physical systems may exhibit non-monotonic properties under certain conditions. The third FoM considered was the ability to handle higher order problems. Many "real-world" systems exhibit highly non-linear and higher order features, especially when examining large ranges in their input variables. Additionally, some distribution approximation methods' simplifying assumptions make them unable to capture this well. The fourth and final FoM identified was the scalability of the method, both with respect to the amount of time required to implement by adding an additional input variable or response and the quality of the representation when the dimensionality of the problem increases. The results for this qualitative comparison are provided graphically in Figure 16.

Figure 16 contains qualitative ranking information for both additive sampling techniques (first 5 rows) as well as distribution approximation techniques (last 3 rows). When considering the additive sampling methods, MC sampling scores well for all parameters except for time cost. This intuitively makes sense, because it is considered the "gold standard" for additive sampling methods, but it is not often used because of the immense time cost associated with creating sufficient samples. All other additive sampling methods are created with the intent of approximating MC sampling results, but doing so with fewer samples. The most promising of these is QMC sampling. It has a lower time cost than all other sampling methods, but as shown, it does not have a "good" score for scalability. This occurs because of the "banding" which occurs for high-dimensional samples (greater than 20 dimensions). However, if fewer than 20 dimensions are considered, this is the superior method. If considering more than 20 dimensions, one should instead select stratified MC sampling, specifically LHS as it has been proven to perform well even for high dimensional

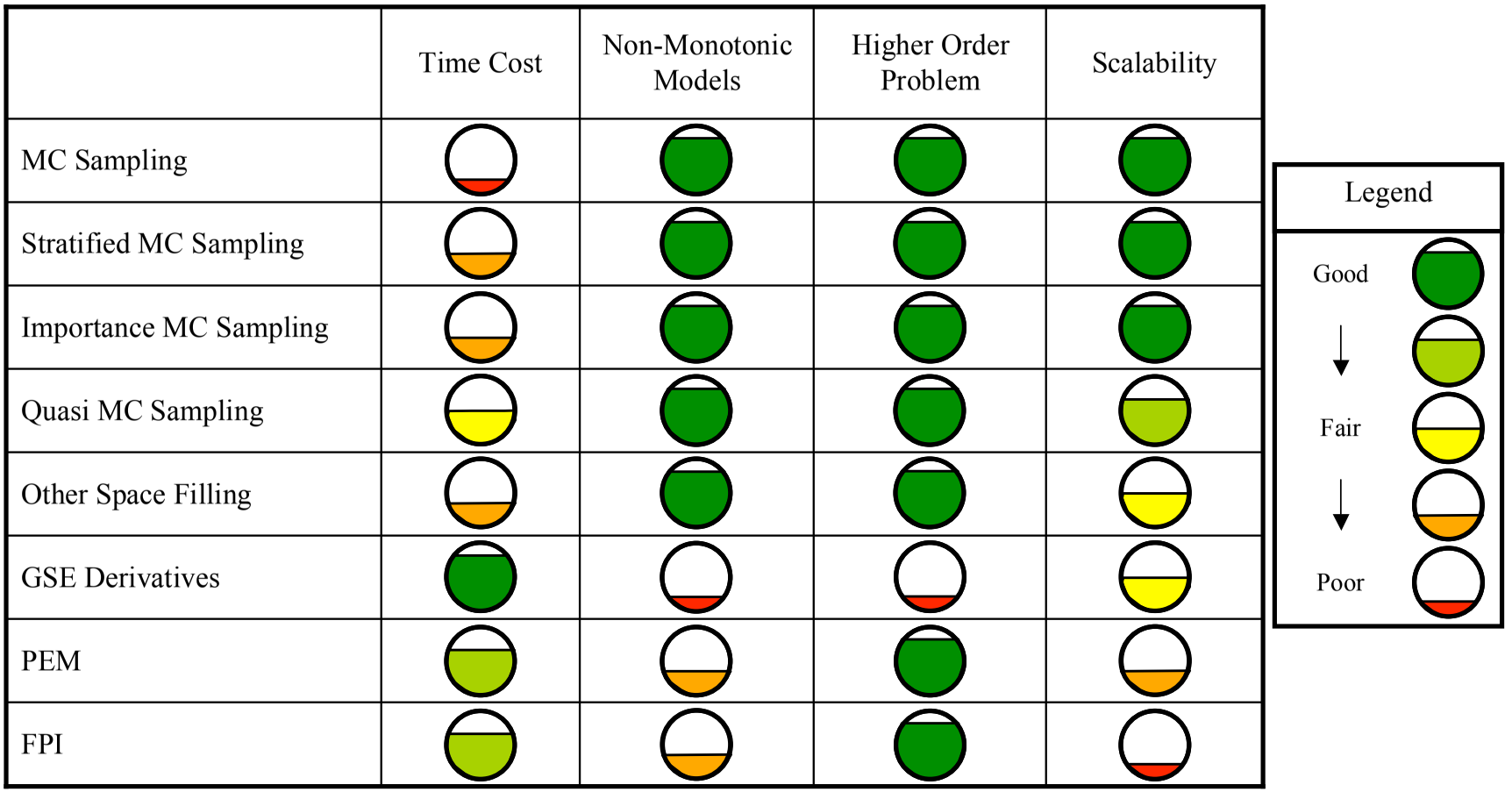


Figure 16: Qualitative Comparison of Uncertainty Propagation Methods.

problems.

Moving on to distribution approximation methods, all of these techniques have a lower time cost than the additive sampling methods. This intuitively makes sense because this was the motivation behind their development. However, they do not perform as well for the other selection criteria. Because of their lower time cost, they should be selected when possible over additive sampling methods, but this is only the case for low dimensional problems. The lowest time cost method is the GSE derivative, but this unfortunately contains linearized assumptions. For this reason it has only limited applicability for small perturbation problems or can only be applied to linear systems. Neither of these are common characteristics for complex physical systems or experiments intended to be conducted for this study, so this method should be discounted. When comparing PEM and FPI, both methods perform well, but PEM scales slightly better than FPI as the number of dimensions increase, and therefore it is the most preferable distribution approximation method.

Based on the findings in Figure 16, three different methods should be selected for further use and one should be selected over another depending on the specifics of the problem being addressed. Three methods should selected because based on this literature review of these methods, three different methods were found to outperform all others, but this performance is dependent upon the situation. The first situation, when the problem is low-dimensional and monotonic, PEM should be used. PEM was found to perform very well for low dimensional monotonic problems. However, if problem is not both monotonic and low-dimensional, one should instead choose a QMC Leaped Halton sequence. This sequence performs well regardless of monotonicity, but regrettably breaks down when more than 20 dimensions exist for the problem. If there are more than 20 dimensions to the problem, one should choose an LHS. This sequence does not suffer from the "banding" issue which breaks down a QMC Leaped Halton sequence and outperformed the other alternatives for these high-dimensional problems.

CHAPTER IV

EVIDENCE THEORY IMPLEMENTATION FOR EPISTEMIC MODEL UNCERTAINTY QUANTIFICATION

4.1 Introduction

In Chapter 2, a case was made for using evidence theory to quantify the epistemic model uncertainty associated with tools in an Modeling and Simulation (M&S) environment. Evidence theory was chosen because of the soundness of the theory, the relative clarity in its implementation as well as the ability to compare results of an evidence theory uncertainty study to those of a probabilistic uncertainty study. Because evidence theory is in essence a generalized form of probability theory, many methods which have been identified and successfully used to quantify uncertainty in probabilistic uncertainty studies can be used for evidence theory uncertainty studies. Chapter 3 surveyed various methods which have been used to quantify probabilistic uncertainty. This chapter is intended to show how these methods can be leveraged for evidence theory studies. First, the case will be made for why an approach containing any possible distribution will provide a more conservative estimate for uncertainty (and in the opinion of the author a more realistic representation of the uncertainty present) relative to a maximum entropy approach commonly taken in probability theory. Next, the specifics for quantifying uncertainty will be outlined. The second challenge with using evidence theory for uncertainty quantification, specifically the immense computational expense relative to a probabilistic uncertainty study, will be confirmed. Hypotheses will be made as to how this issue can be resolved and experiments will be outlined to test this hypothesis. The results from these experiments will be shown to support this hypothesis. Guidelines for future studies quantifying epistemic model uncertainty using evidence theory will be created for future use in this document.

4.2 Identifying Bounds with Any Possible Distribution

Using an evidence theory approach for uncertainty quantification differs from approaches documented in the literature for probabilistic uncertainty quantification. In a probabilistic study, a distribution type is identified through available data, subjectively selected. These probabilistic approaches will propagate this specific distribution shape through the system of interest; and confidence in the resulting outputs can be assessed using a Cumulative Distribution Function (CDF) of the outputs of interest. An evidence theory formulation is more general, and as Dempster intended, this should represent any possible distribution type. In surveying the literature, methods for conducting this evaluation are either simplistic and monotonic [169], questionable in their specific implementation[102], or vague on how computational expense is addressed[3].

Monotonic problems were considered in the "challenge problems" set identified by Sandia Labs. While it is clear that the diversity of possible outcomes can be obtained though sampling all possible combinations of end-points for a simple monotonic system, the combination of inputs which produce the maximum variation in the output are significantly less clear for engineering problems which are non-monotic.

Later work done at Sandia Labs chose to take a sampling-based strategy.[102] Here the authors used an Latin Hypercube Sampling (LHS) to sample the space. In probability theory applications, the creators of LHS showed that these samples, though uniform, could be considered a "blank" distribution.[112] This would allow biasing to be applied to the various samples, and one could use a single set of samples, apply various biasing schemes to these samples, and then the results would represent a variety of different distributions. This author believes this approach, though valid for a probabilistic sampling, is not valid for an evidence theory formulation. While this biasing may represent some small change in the results, it would not increase the probability of having samples in the corners of the input space, meaning the likelihood of having all high settings on variables or all combinations of high and low settings of input variables is extremely unlikely even when using an LHS to better distribute the samples (this will become more clear later in this chapter). For this reason, it is the opinion of the author that this approach is questionable and it will not be

pursued any further.

The third approach for using evidence theory in the literature commonly involves the authors giving only a very vague description of how any possible distribution would be represented.[3] Based on these findings, there is a clear need for a more traceable process for using evidence theory to quantify uncertainty.

4.2.1 Uncertainty Quantification Research Questions and Hypotheses

This author was unable to identify any sources in the literature where the following two research questions for an evidence theory formulation for representing uncertainty were answered.

1. How should one strategically create these distributions to represent any possible distribution as efficiently as possible?

2. How many distributions should be used to represent any possible type of distribution?

The first research question was addressed with a literature review of areas separate from the evidence theory community. After the literature review was conducted, an example was created to illustrate the feasibility of the answer chosen and illustrate how the distributions could be strategically created. This literature review, its findings, and the example are provided in the following section. The second research question, which asks how many distributions are needed to represent any possible distribution, led to the following hypothesis: that **the number of distributions required to capture the maximum possible variability in a response will asymptotically approach the number necessary and this number will be dependent on the specific problem and the number of inputs being examined.** This hypothesis is illustrated in Figure 17 and will be tested after the first experiment.

4.3 Distribution Creation Findings

After a literature review, three different types of distributions were considered:

1. Beta Distributions

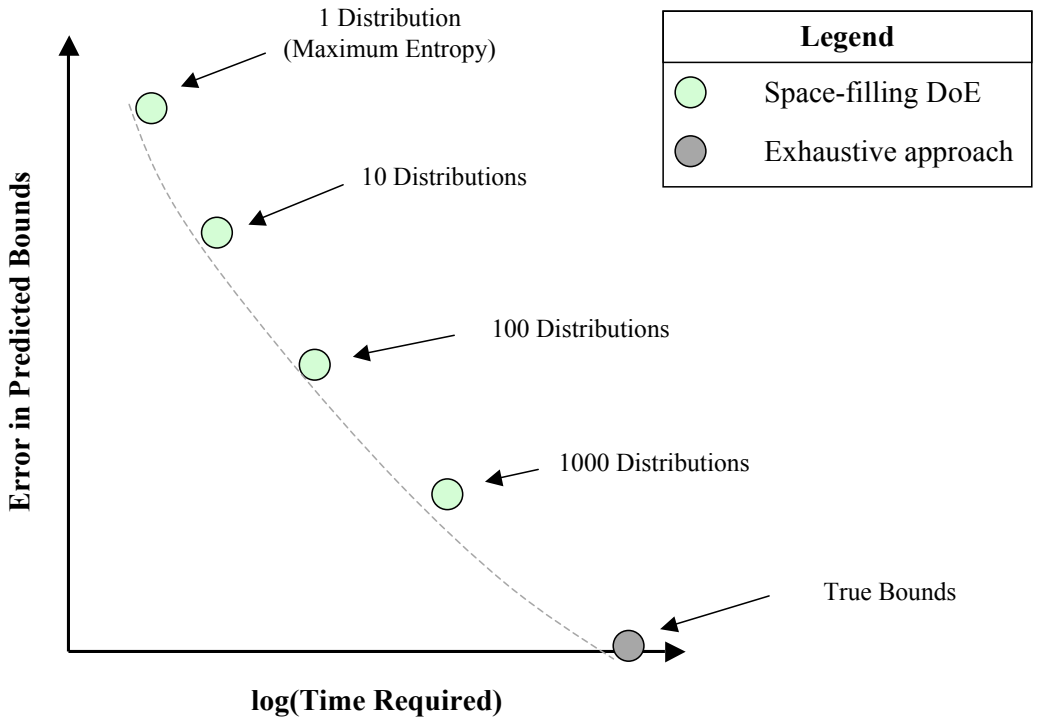


Figure 17: Notional Results Hypothesized for Experiment 1.

2. Johnson Functions

3. Custom Distributions

Custom distributions were dismissed because it would require a significant amount of "tuning" and while the Probability Density Function (PDF) could be created for any, arbitrary Cumulative Distribution Function (CDF), this would require a solver, as was discussed in Chapter 3, which would require more time. Johnson functions were also considered, but these were dismissed because they are not as readily available to future users. Beta distributions were chosen because they are able to easily create a variety of non-infinite distributions without using an external solver to obtain the PDF and because they are readily available in many most software packages. A standard beta distribution is defined in the interval [0,1]. A standard beta possesses two shape parameters, a and b; the closed form expression for a standard beta distribution PDF is provided in Equation 19.

$$f(x) = \frac{x^{a-1}(1-x)^{b-1}}{\int_0^1 u^{a-1}(1-u)^{b-1}du} \tag{19}$$

Where:

- $x \in [0, 1]$
- a,b are standard beta shape parameters

Standard beta shape parameters can be used to create a variety of different distributions. Specifically, if they are equal, the distribution is symmetric. If $a > b$, then the mean is right of center. If $a < b$, then the mean is left of center. If $a = b = 1$, then the distribution is uniform. If either a or b are less than unity, then the corresponding side is extreme; otherwise it is not. Figure 18 is provided to illustrate some of the diversity of standard beta distributions. It should be noted that in the literature, standard beta distributions are also referred to as two-parameter beta distributions. In some sources, authors reference using four-parameter beta distributions.[217] These are simply two-parameter (standard) beta distributions, but scaled over some range. The third and fourth parameter are the lower and upper bounds. A standard beta distribution represented with four parameters would be $beta(a, b, 0, 1)$. This scaling can be achieved through simple multiplication and will be left to the reader if they wish to further examine the differences.

After identifying a promising candidate distribution type, a method for strategically sampling these must be identified. As discussed in Chapter 3, the most promising sampling method when sampling fewer than 20 input dimensions is a leaped Halton Quasi Monte Carlo (QMC) sequence. Because a standard beta only has two parameters, it meets this criteria. However, because there are only two dimensions for this problem, leaping is not necessary to resolve "banding" issues. Due to the fact that standard beta distributions have been used by many others in the literature with success to represent a great variety of non-infinite distribution shapes[217, 87], no other alternatives were considered. While beta distributions can be used for creating a great variety of different shapes, the question remains as to what parameters should be strategically varied with QMC sequences. Two possible scenarios were considered:

1. Varying standard beta mean (μ) and standard deviation (σ)

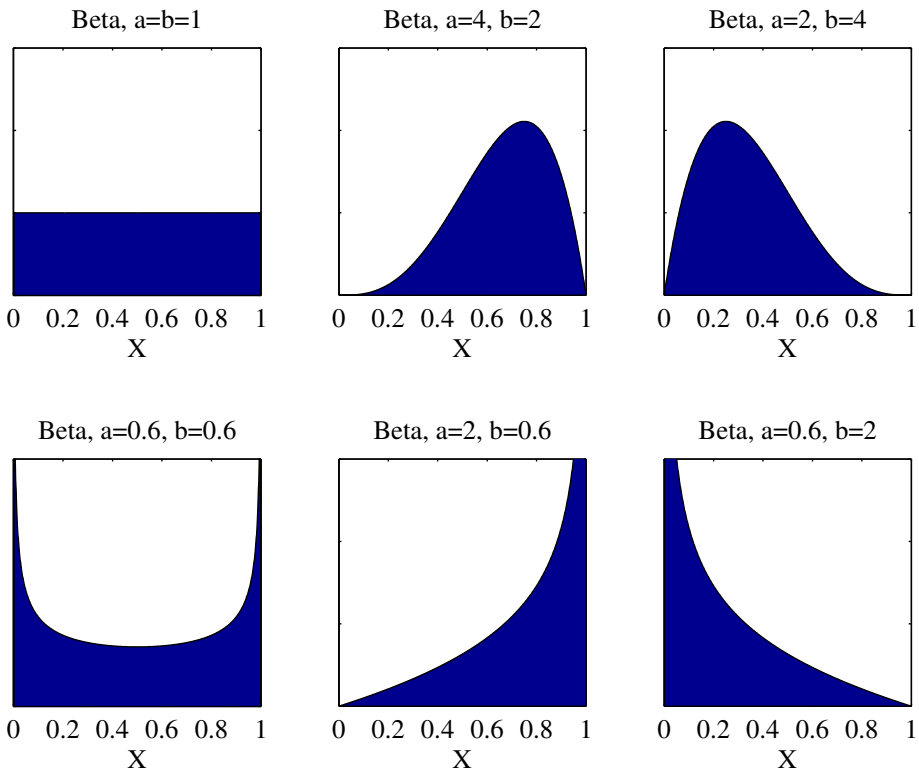


Figure 18: Illustration of Standard Beta Distribution Shape Diversity.

2. Varying standard beta shape parameters a and b

The first scenario was thought to be a better way for representing a variety of different distributions. This was considered to be better because the mean and standard deviation for an arbitrary distribution are physically realized and would represent a wide variety of different shapes. In implementation, however, several issues were identified. For most combinations of means and standard deviations, a beta distribution could be created. However, when considering extremely high or low means and large standard deviations, a beta distribution could not be defined. For this reason, the first scenario was discarded.

The second scenario was examined and was found to out-perform the first scenario. While parameters such as the mean and standard deviation could not be controlled as independent parameters, varying standard beta distribution shape parameters a and b yields better results than the first. Using the second scenario, the following scheme was found to produce a good variety of distributions: a and b were each varied from 0.6 to 6.0, but were

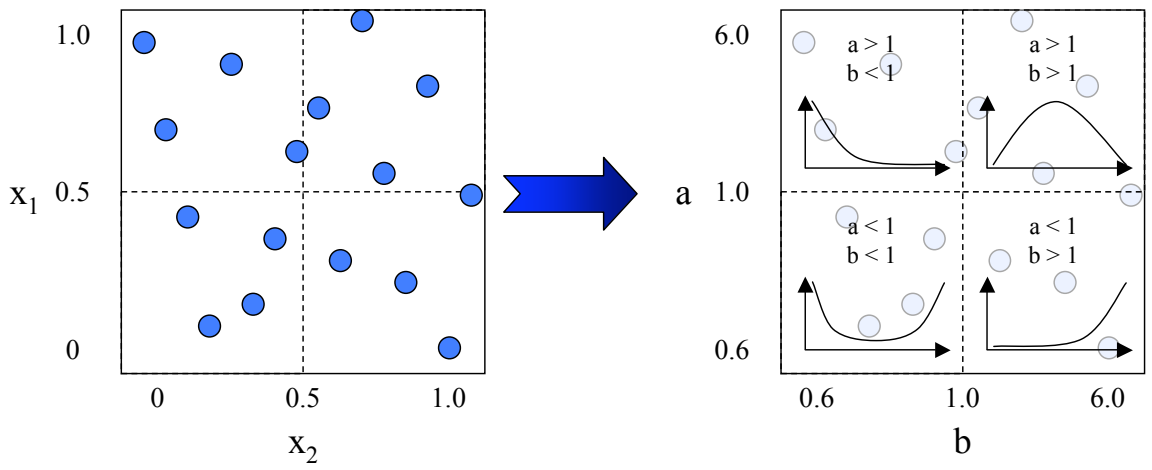


Figure 19: Illustration of Transformation from QMC Samples to Standard Beta Distribution Shape Parameters.

centered on 1.0. This would produce a scenario where half of the created distributions were extreme on the left side, half were extreme on the right side, one quarter were extreme on both sides, and one quarter were not extreme at all. The transformation from a QMC sample to the proposed approach is illustrated in Figure 19. In this figure, 16 QMC samples are taken. These samples are then transformed to the beta shape parameter space producing 16 different beta distribution shapes. Given a uniform LHS sample, one could perform an inverse transformation (as was discussed in Chapter 3 for Importance Sampling (IS) applications) to change the shape of this uniform distribution to that of a beta distribution with the desired shape. The resulting beta distribution can then be analyzed in some analysis of interest in an effort to capture the bounds of the space.

An example of how this method can quickly generate a great variety of different distributions is provided in Figure 20. This figure graphically shows how a good variety of distributions can be created using this method. It should be noted that one should include at least six distributions in a given sample. This is due to the nature of a QMC sampling, which is used to create the beta distributions. As discussed earlier in Chapter 3, a Halton QMC sequence must complete its radix to have good space filling properties. This radix is the product of the primes used for each dimension. In this implementation, there are two dimensions, (a and b). The first two primes in a Halton QMC sequence are 2 and 3,

so the sequence must have $2 \times 3 = 6$ samples. Alternatives to using a QMC sequence for fewer than six samples were considered, including LHS. An LHS does outperform a QMC sequence for fewer than six samples, but neither has good space filling properties. If one must propagate fewer than six beta distributions, it is recommended that they either be manually selected or generated with a method such as Optimal Latin Hypercube (OLH) [177].

This beta distribution creation method adequately creates a great variety of different distribution shapes and will be used in the subsequent experiments.

4.4 Experiment 1

A first experiment was conducted to identify how many different distributions are necessary to capture the variability in an output caused by any number of possible distributions. It was hypothesized that the results for such an experiment were notionally illustrated in Figure 17. The distributions are created using the Halton QMC sequence outlined in the previous example.

The first and most obvious method for selecting distributions is to do so with a full-factorial, meaning every combination is compared with every other combination. A scheme such as this, though extremely thorough, is also extremely computationally expensive. Using such an approach, the number of total distribution combinations evaluated would grow as the number of different distributions raised to the number of variables. For example, if one were to consider 3 different variables and used 10 different distributions, this would require evaluating $10^3 = 1,000$ distributions. If using a vector-oriented simulation package, such as MATLAB, this would require $1,000$ function evaluations. If one were instead using a more traditional software package and each sample was evaluated individually, this would require $1,000$ multiplied by the number of samples per distribution. If there were 100 samples in each distribution, this requires $100,000$ function evaluations. With $1,000$ samples in each distribution it requires $1,000,000$ function evaluations. While this scheme quickly becomes computationally infeasible, especially when considering large problems, it should be assessed in order to test the hypothesis posed earlier. Additionally, with computing parallelization

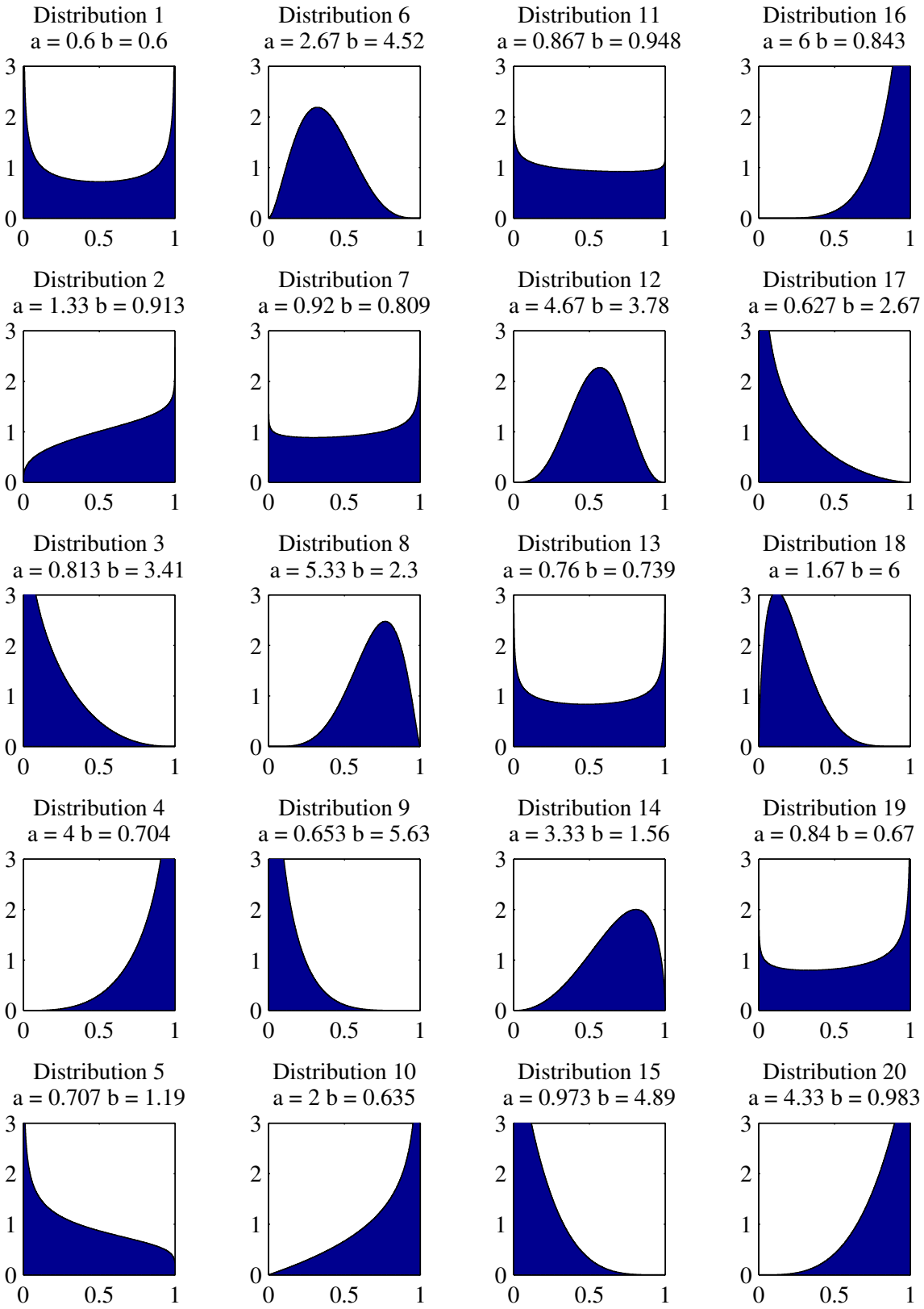


Figure 20: Illustration of First Twenty Beta Distributions Created in the Selected Distribution Creation Scheme.

and efficient implementation, these calculations may be performed more quickly. In order to test how many distributions would be required when using a full-factorial distribution combination, three test problems were identified and examined.

4.4.1 Test Functions

Three test functions were identified in the literature for testing uncertainty hypothesis 2. These functions are discussed here with respect to the number of possible input variables, ranges for input and output variables, and their 2-D shape. The three selected functions differ in ranges for input and output variables, relative location of global maximums and minimums, and overall complexity, as will be illustrated in the subsequent subsections.

4.4.1.1 Schwefel Test Function

The first function considered was the Schwefel test function. This function is an n-dimensional problem whose closed-form expression can be found in Equation 20. A surface plot of the Schwefel function for $n = 2$ is graphically provided in Figure 21. This test function was selected because it is n-dimensional, because it has a large variation in both the input variables and output variable, and because of the complexity of its surface. Its n-dimensional nature allows for possible trends with increasing number of input dimensions to be captured. The sinusoidal nature of this function means it has several peaks, and as illustrated in Figure 21, it has four global maximums and four global minimums. The reader should note that these, much like the maximum and minimum for many engineering problems, are located at the edges of the space rather than in the center.

$$f(x) = 418.9829n \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}) \tag{20}$$

Where:

- n is the number of input variable dimensions.
- $x \in [-500, 500]$

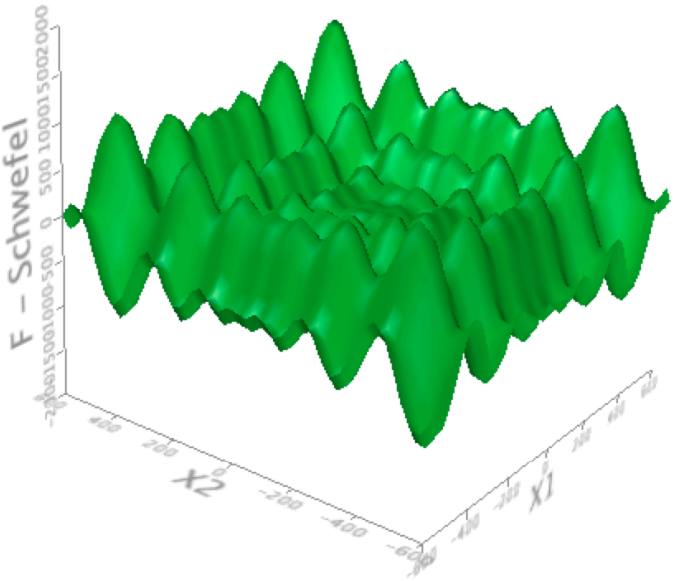


Figure 21: Schwefel Function Surface Plot.

4.4.1.2 Michalewicz Test Function

The second test function selected is the Michalewicz test function. This n-dimensional model is provided in Equation 21. A surface plot of the Michalewicz function for $n = 2$ is provided in Figure 22. This test function was selected because it is an n-dimensional function, because of the relatively small variation in both the input variables and output variable, and because of its complexity and extreme "peakiness" of the global maximums and minimums. This space is characterized by large flat spaces with steep, nearly discontinuous changes occurring between these flat spaces. One should note that, like the Schwefel function and many engineering applications, both the global maximums and minimums occur at the edges of the space.

$$f(x) = (-1)\sum_{i=1}^{n} \sin(x_i)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m} \tag{21}$$

Where:

- n is the number of input variable dimensions.
- m is an arbitrary constant, for this implementation, $m = 10$
- $x \in [-3, 3]$

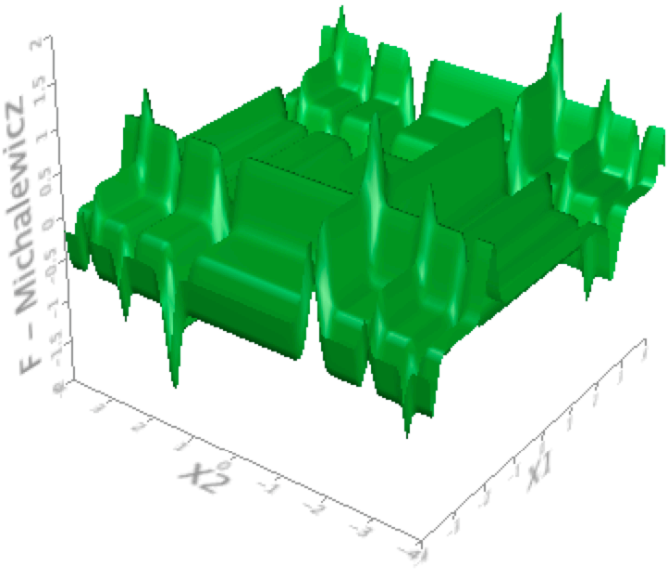


Figure 22: Michalewicz Function Surface Plot.

4.4.1.3 Sphere Test Function

The third and final test function identified for this experiment is the Sphere test function. This function is n-dimensional like the other two, and its closed form expression is provided in Figure 22. This function is significantly different from the other two test functions and was selected because of its relative simplicity. It should test whether it is possible for the bounds of a relatively simple function to be captured using a maximum entropy approach which would use a single (uniform in this case) distribution. A surface plot, illustrating the simplicity of the Sphere test function for $n = 2$, is provided in Figure 23.

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{22}$$

Where:

- n is the number of input variable dimensions.
- $x \in [-5, 5]$

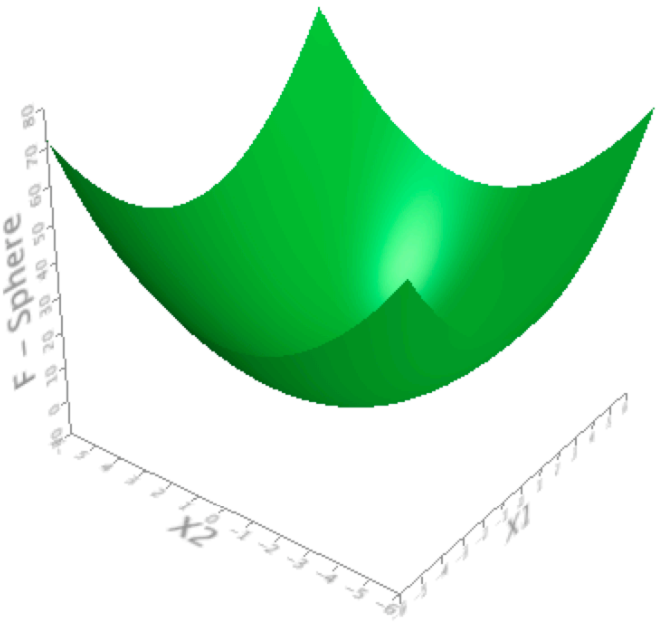


Figure 23: Sphere Function Surface Plot.

4.4.2 Experimental Setup and Test Function Results for Full Factorial Distribution Combination

To test this hypothesis, the following approach was used to create a variety of distributions and then calculate the resulting bounds:

1. Determine the number of different distributions analysis
2. Identify the beta shape parameters for each of these different distributions
3. Determine the number of input variables
4. Create a uniform sample for each input variable
5. Perform the inverse transformation for each input variable and distribution combination
6. Evaluate each sample in the distributions for this case
7. Determine the bound(s) in the output(s) of interest for this case
8. After evaluating all cases, determine the bound(s) captured in the output(s) of interest

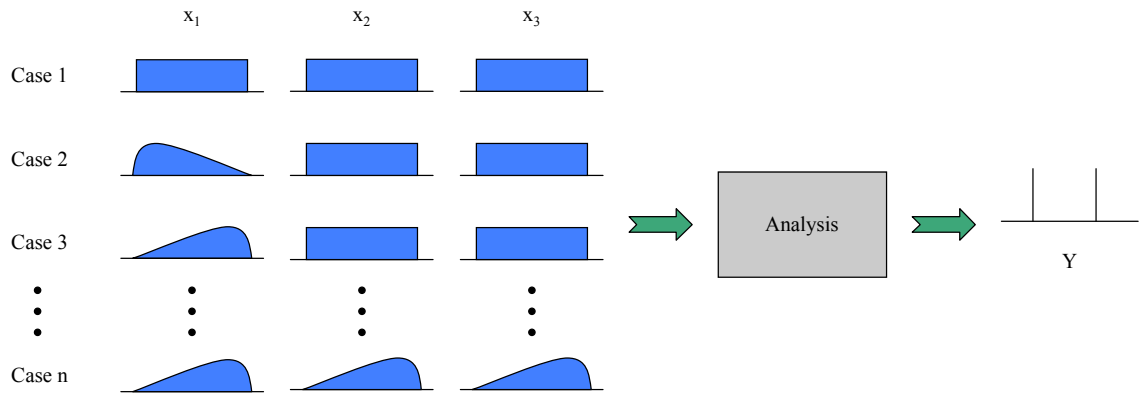


Figure 24: Illustration of Sampling Process for a Full Factorial of Distribution Shapes to Capture The Bounds of Space.

For this experiment, the number of different distributions in item 1 was varied. The process for item 2 is described earlier in Figure 19. This process involves using a Halton QMC sequence to sample the beta parameter shape space, once for each different distribution considered. The number of input variables (item 3) was also varied for this experiment. The uniform sample created for each input variable in item 4 was done using a Halton QMC sequence. The number of samples that were taken in this Halton sequence was also varied in the experiment. While the "banding" is a concern for higher dimensional problems, it not significant for low dimensional problems (which are being considered here), and therefore no leaping was necessary. The next step, which involves performing an inverse transformation of each variable and distribution combination was performed using the Matlab built-in function. The function being evaluated in item 6 was varied, and all three of the optimization test functions were considered. Following this evaluation, the bounds of the space could then be calculated for each case in the full factorial and once all of the full factorial cases had been evaluated. This process is illustrated graphically in Figure 24.

This experiment was intended to identify the asymptote in the bounds captured with respect to the number of input variables. This was done for all three test functions. Additionally, the number of samples in the distributions were varied. This was done in an effort to determine whether it would be better to place more samples in a few different distributions or to place only a few samples in many different distributions. The first parameter of

interest, the number of input variables, was varied from 2 to 4. While data for more than 4 input variables would have been of interest, it was computationally infeasible to generate it. Generating data for 4 input variable dimensions took nearly one day using vectorized MATLAB. The number of points per distribution was examined at 100, 1,000, and 10,000.

One should notice that when quantifying uncertainty with the "adder" approach discussed earlier in Chapter 2, the input is the value and bounds of the "adder" function. This differs from the n-dimensional optimization functions selected here. When applying the "adder" to the result of an analysis, the bounds can be very simply calculated as this is a monotonic problem. If, however, the result after the "adder" is applied then passes into another analysis tool before the output of interest is calculated, one could also think of this "adder" to be an input to the later analysis. Therefore, this experiment serves to replace the downstream analysis with an n-dimensional optimization test function. This allows for the experimental results to identify possible trends with respect to dimensionality and this takes advantage of more commonly used optimization functions.

The results for the Schwefel test function are provided in Figure 25. In this figure, the number of different distributions evaluated is plotted against the percent of the relative bounds captured. For a test function such as this, the true bounds of the space can be solved. However, this is not the case for most engineering applications. For this reason, the true bounds were considered to be the difference between the maximum and minimum of any samples from any number of distributions evaluated.

When examining Figure 25, one should notice that for only two input variables, the true bounds of the space are captured relatively quickly, with a relatively small number of different distributions. As the number of input variables increases, it requires consecutively more different distributions to capture the entire bounds of the space. A second significant trend is that an asymptote occurs. This trend supports the hypothesis made earlier. The third trend of significance in this plot is with relation to the number of points per distribution. Regardless of the number of input parameters, more information is gained by evaluating more points in a given distribution. This, however, does not answer the question of whether it is more cost-effective to evaluate a large number of points in only a few distributions or

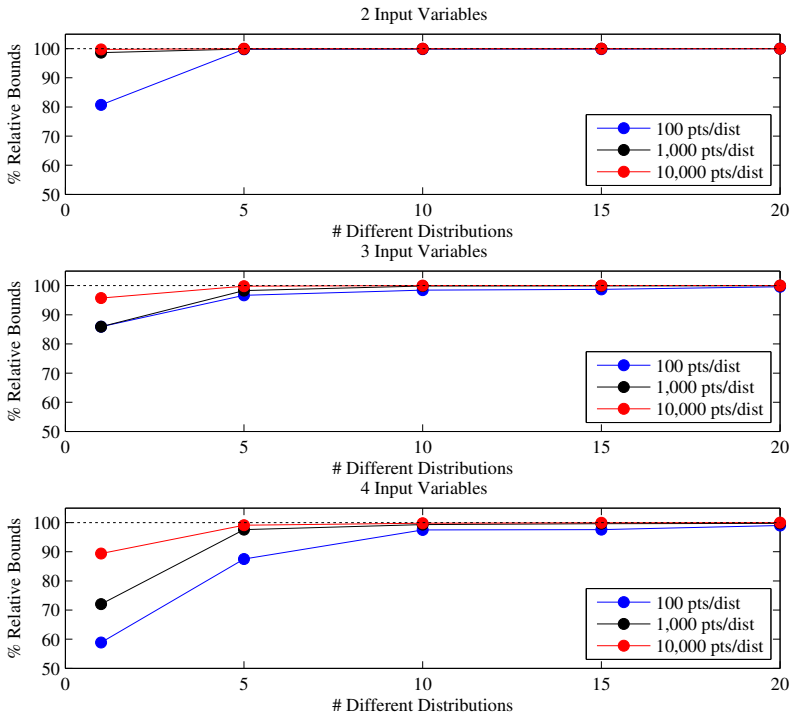


Figure 25: Schwefel Function Percent Relative Bounds Captured vs. Number of Different Distributions for a Full Factorial of Different Distributions.

to evaluate only a few points in a large number of distributions. To examine this trend, the reader is directed to Figure 26. Here the reader can see that the results are inconclusive. While for certain numbers of input variables there is separation between the red, blue, and black lines, in others there is not. To further examine this trend, the reader should consider data from other test functions.

Moving on to the Michalewicz test function, Figure 27 contains the number of different distributions plotted against the percent of the relative bounds captured. Here, the percent relative bounds are calculated based on the maximum bounds captured by any evaluation from any number of distributions as they were for the Schwefel test function results. As with the Schwefel test function data, a clear asymptote is seen as the number of different distributions increases. There is also a clear shift in this asymptote to the right as the number of input variables increases.

To further consider the trade-off between the number of different distributions evaluated and the number of points in a given distribution, the reader should consult Figure 28. As

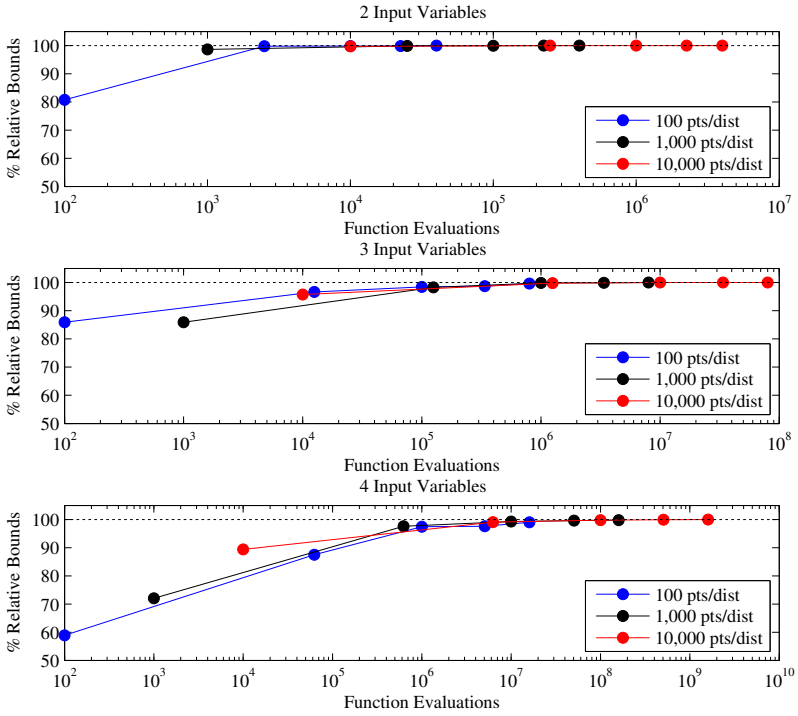


Figure 26: Schwefel Function Percent Relative Bounds Captured vs. Number of Function Evaluations for a Full Factorial of Different Distributions.

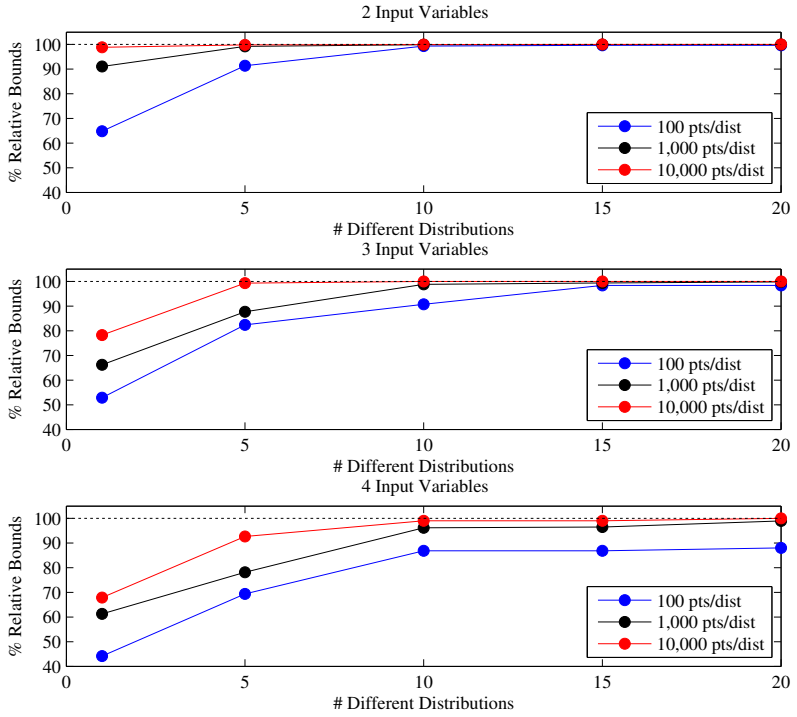


Figure 27: Michalewicz Function Percent Relative Bounds Captured vs. Number of Different Distributions for a Full Factorial of Different Distributions.

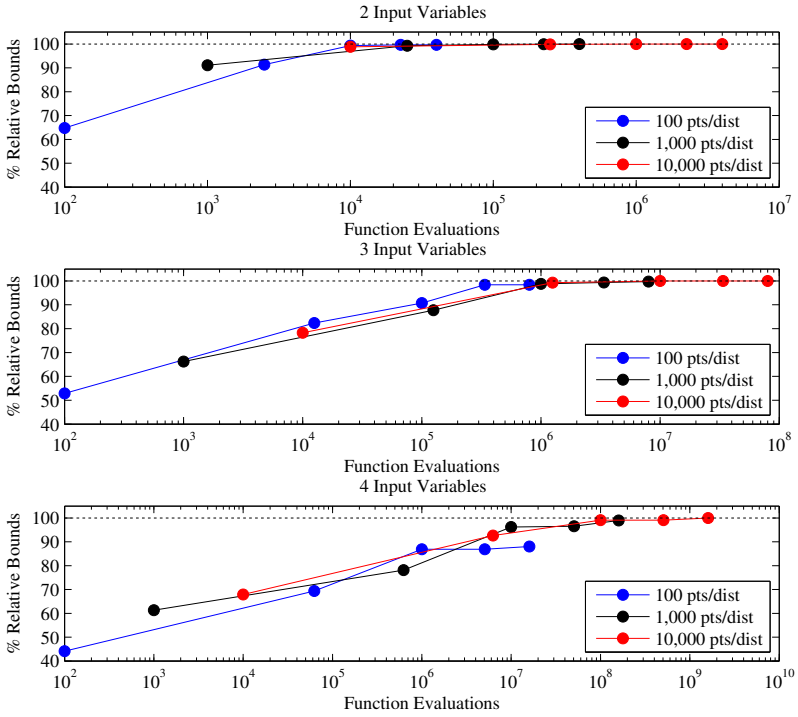


Figure 28: Michalewicz Function Percent Relative Bounds Captured vs. Number of Function Evaluations for a Full Factorial of Different Distributions.

was the case when examining Figure 26, there is no clear benefit to having either more points in a distribution or more different distributions evaluated.

When considering the third and final test function, the Sphere test function, the reader should first consult Figure 29. In this figure the reader sees trends much like those seen in Figures 25 and 27, though these trends are shifted to the left relative to the other test problems. Nearly 100% of the relative bounds are captured with a single distribution for 2 input variables, and it requires only slightly more different distributions as the number of input variables increases. As was mentioned earlier, this confirms the belief that maximum entropy can capture the bounds of certain problems. However, when re-examining the surface plot for the Sphere test function provided in Figure 23 and those of either the Schwefel or Michalewicz test functions in Figures 21 or 22, respectively, one should note that these functions differ *significantly* in complexity both of the surface and in the relative area around the global maximum and minimum. For this reason, a maximum entropy approach should generally not be considered to capture the bounds because of its poor

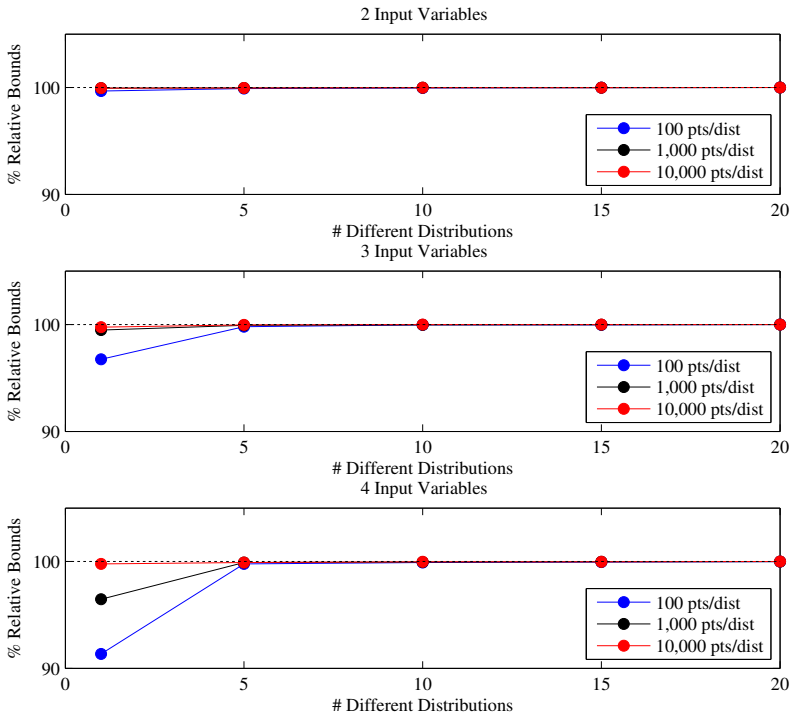


Figure 29: Sphere Function Percent Relative Bounds Captured vs. Number of Different Distributions for a Full Factorial of Different Distributions.

performance on any problems other than extremely low-dimensional, simple problems.

Re-examining the trade-off between the number of points in a distribution and number of distributions evaluated, the reader should examine Figure 30. This figure reiterates information provided in Figures 26 and 28. There does not appear to be a clear benefit in the speed that one can capture the bounds when trading off between the number of points in a distribution and the number of different distributions. One should note, however that all these bounds are obtained through a relative asymptote. If one were to implement an asymptote-finding algorithm, it would terminate after the bounds remains unchanged for a given number of different distributions. The difficulty would be in ensuring the convergence rule was stringent enough. If it was too lenient, it would terminate too soon (before the asymptote had been reached). If it was too stringent, the method would take more samples than necessary and time would be wasted.

Throughout these results, there has been a clear benefit in considering different distributions relative to a signal distribution as would be the case from a maximum entropy

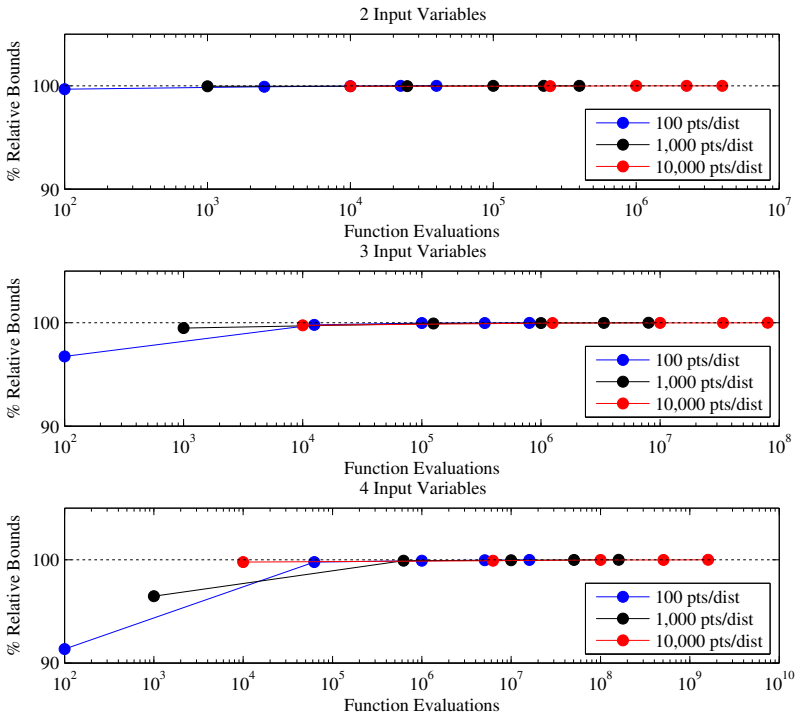


Figure 30: Sphere Function Percent Relative Bounds Captured vs. Number of Function Evaluations for a Full Factorial of Different Distributions.

approach. The only exception to this was for the simplistic Sphere test problem and only when considering two input parameters. While the trade-off between number of samples in a distribution and number of different distributions was considered, there was always a benefit in the Schwefel and Michalewicz functions to consider more than one distribution (and in the Sphere as well for more than 2 input variables).

The reason for this can be explained with Figures 31 and 32. In Figure 31, a simple function was analyzed with a series of different input variables. The function analyzed is provided in Equation 23. Each of the inputs for this function, x_i were taken to be random variables, where $x_i \in [0, 1]$. This represents a maximum entropy approach to capturing the bounds of the space.

$$Y = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{23}$$

Where:

- n is the number inputs

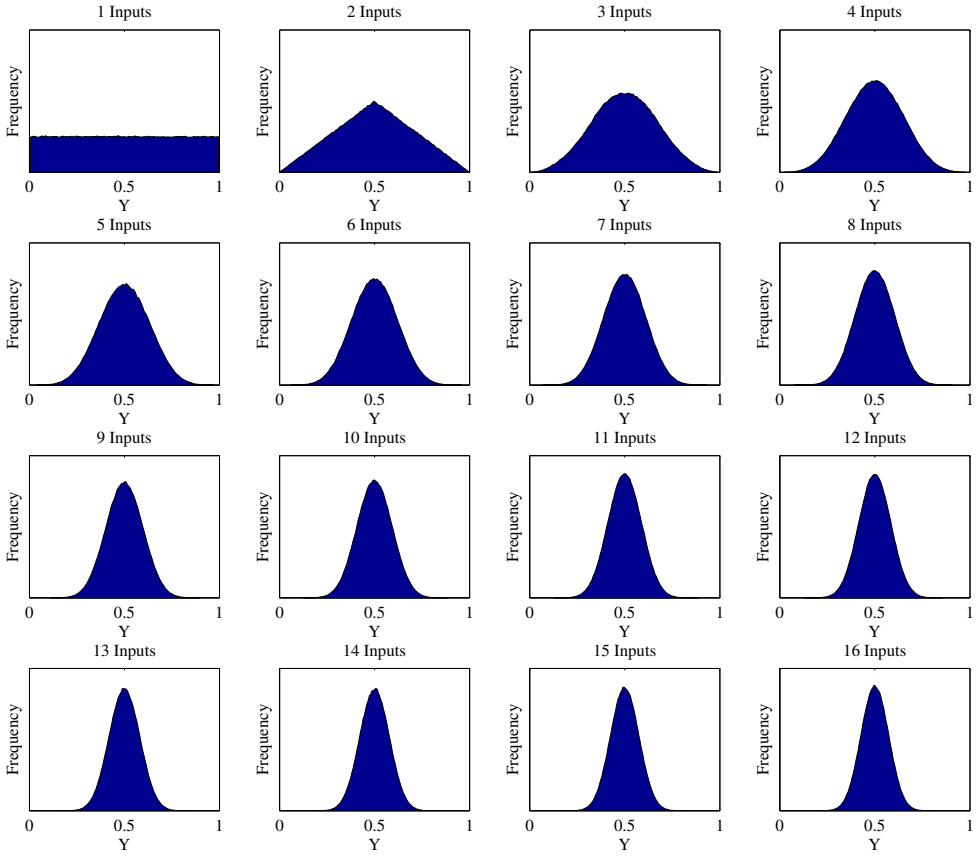


Figure 31: Histograms for Average Function Result for Various Input Variables.

As one can clearly see in Figure 31, as the number of inputs increases, the distribution approaches a normal distribution. This is what would be expected and is essentially illustrating the Central Limit Theorem [96]. When the goal of this sampling is to capture the bounds of the space, this means that it becomes increasingly unlikely that one would capture the bounds of the space. When looking at the histograms provided in this figure, the obvious conclusion should be that a maximum entropy approach is not an effective means for capturing the bounds of the space. Advocates for a maximum entropy approach for capturing the bounds of the space would argue that the mapping shown here is not applicable to other problems (such as theirs) and that the mapping from the random variables to the output is strictly a function of the problem being addressed. The author would agree with this, but through simple experiments one can easily show that multiplication and division cause uniform distributions to coalesce into a normal distribution (or at least centered, "peaky" distribution) much more quickly than an operation like addition or subtraction. Given this simple experiment, one can easily see that regardless of the operation being performed to obtain the input, it is *extremely* unlikely the bounds will be captured with a maximum entropy approach. The reason for this is that as the number of inputs increases, the likelihood of having the appropriate settings occur simultaneously, though possible, is extremely unlikely. This is why the approach proposed here outperforms a maximum entropy approach. Rather than having a uniform distribution over some bound, this approach shifts the "focus" of the distribution from the center of the space to the upper and lower sides, and this in turn increases the likelihood of having the appropriate high/low settings occur simultaneously.

The second argument presented by advocates of a maximum entropy approach to quantifying the bounds or the entire design space is that by increasing the sample size, one can in effect "beat" the Central Limit Theorem and still adequately capture what is occurring at the extremes of the output space even though there is a low probability of samples falling in this region. In order to test this, the same test problem was used and the total number of samples which fall at the extreme 10% of the space were calculated, meaning a sample would be counted only if $Y \in [0.0, 0.1] \bigcup [0.9, 1.0]$. The result from this test is provided in

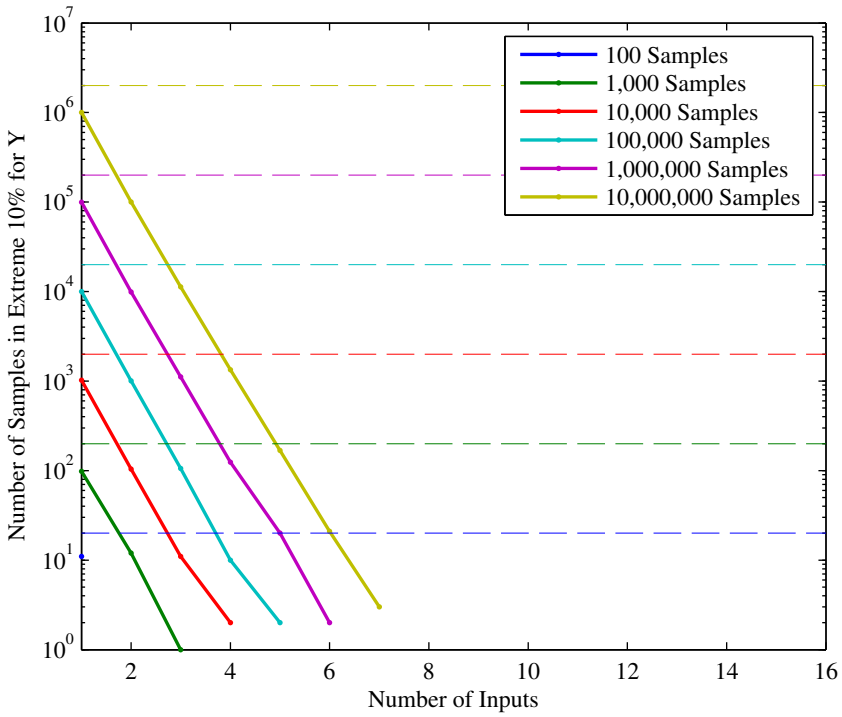


Figure 32: Number of Extreme Samples from Average Function for Various Numbers of Inputs and Sample Sizes.

Figure 32. As seen in Figure 31, the distribution has fewer points at the extremes as the number of inputs increases. When counting these points at the tails of the distributions, there is a large amount of variability in the number of samples. In an effort to reduce this variability, the distribution for Y provided in Figure 32 is actually the average of 10 iterations. The intent behind doing this was to smooth the lines in Figure 32, but but there clearly some variability even given this averaging.

In this figure, the results for the experiment are shown with a solid line whose color is determined by the sample size. This is plotted on a semi-log plot, not a linear plot (the vertical axis is a log plot). In order to determine how the number of samples found in this extreme region compare to what would be obtained if the samples were actually truly uniform, a dotted line of the same color has been provided indicating how many samples would be necessary in the extreme region if the extremes of the space were being sampled well. The result in Figure 32 does two things. First, it does partially confirm the maximum entropy advocate's position. The number of samples in this extreme region increases if one

increases the number of samples taken. However, this also shows that while the number of samples increases, the decrease in samples in the extreme region happens exponentially regardless of how many inputs there are. Take for example the limiting case. If only 100 samples were taken, Figure 32 shows the reader that there would not be *any* samples in the extreme 10% of the space if there was more than 1 input variable. Similar trends are seen for each sample size, and interestingly even if $10,000,000$ samples were taken, one should not expect to find samples in the extreme 10% of the bounds if there are more than 7 input variables. It is true that if enough samples were taken, samples would eventually be found at the extremes of the output space, but the number of samples would be computationally infeasible to generate, let alone evaluate with any analysis.

A second important observation from these results is in how much more slowly the method is able to capture the bounds of the space for the Michalewicz function relative to either the Schwefel or Sphere functions. The reason for this is due to the relative area around the global maximum and minimum. Because this is intrinsically a sampling-based approach, the ability to capture the maximum or minimum is dependent on how close samples can be placed to the maximum or minimum. If the function being evaluated is very steep near the optimum, meaning there is a relatively small area where a specific combination of variables must fall to capture the maximum or minimum within some limit, it equivalently means more different combinations must be evaluated until this small area can be found. This is graphically shown in the left side of Figure 33. The Michalewicz function could be considered an extreme version of this type and the Schwefel function could be considered this type of problem as well. If, however, the space around the maximum or minimum was relatively flat, then there would be a larger area where a specific combination of variables must fall to capture the extreme meaning that fewer different types of distributions would need to be examined. This is represented with the right side of Figure 33, and the Sphere test function could be considered an example of this type.

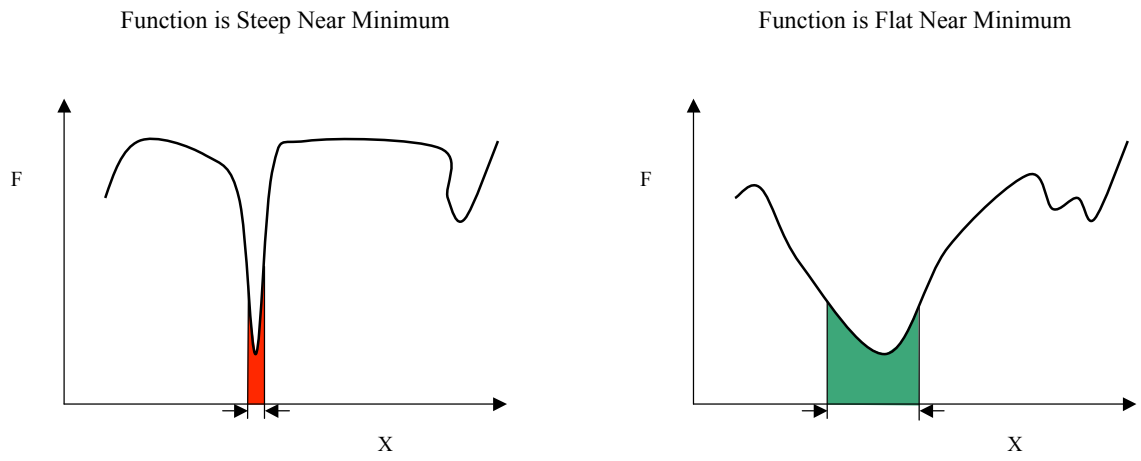


Figure 33: Illustration of Relatively Steep and Flat Functions Near Minimum.

4.4.3 Experiment 1 Conclusions

An experiment was designed to test uncertainty hypothesis 1. The results, which were discussed at length in the previous section, supported the hypothesis. While the hypothesis was supported, an issue has also arisen. The bounds of the space can be captured more effectively with a variety of different distributions, but the computational cost for calculating them quickly becomes infeasible. A problem with four input variables was calculated in approximately one day, and while this may be computationally feasible for uncertainty quantification, the method developed was intended to serve as an input for another analysis which would select analysis tools. For this reason, the resulting method is computationally infeasible. Further approximation is necessary, and will be examined from this point forward. It is expected that a full-factorial, though exhaustive, likely contains a large amount of repetitive information. For this reason, much of this repetition can likely be removed and the bounds may still be captured.

4.5 Alternative Approaches to Full Factorial Distribution Combinations

Two methods were considered for accelerating this process. The first is to use frontier-finding algorithms and the second is to use a space-filling Design of Experiments (DoE) to approximate results from a full-factorial. When considering frontier finding algorithms

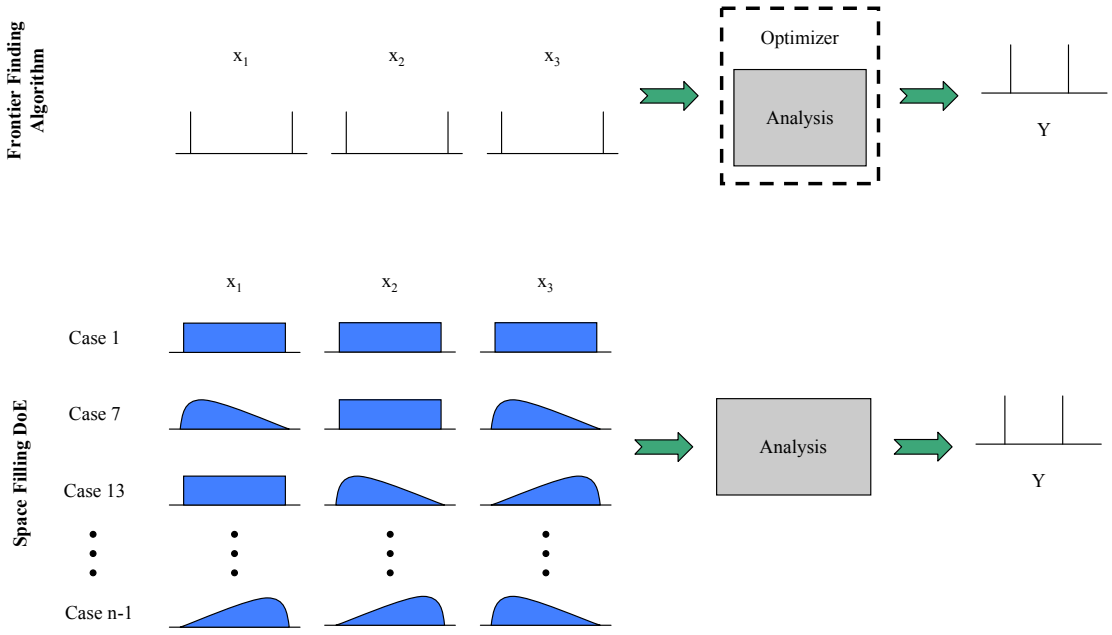


Figure 34: Illustration of Sampling Process for Two Approximations of a Full Factorial of Distribution Shapes to Capture The Bounds of Space.

specifically, a continuous form of Ant Colony Optimization (ACO)[214], a real-coded form of Nondominated Sorting Genetic Algorithm - II (NSGA-II)[50], Particle Swarm Optimization (PSO)[118], and similar methods[26] should be examined. These methods have been used in the literature to identify Pareto frontiers. They could be modified to search for any boundary rather than a single boundary in a given direction. These methods perform relatively quickly, though they have been documented to have some difficulty with non-convex problems. Rather than creating some great variety of distributions to capture the bounds of the space, this approach is an abstraction which instead directly finds the bounds. A graphic illustration of each of this alternative is provided in Figure 34. One should note that unlike the full factorial distribution case illustrated in Figure 24, here the bounds for the input variables are specified, not a series of cases where each is a different distribution shape combinations. Here an optimizer is used to directly solve for the bounds of the space given the specified variable ranges instead of running a series of distributions to find the bounds.

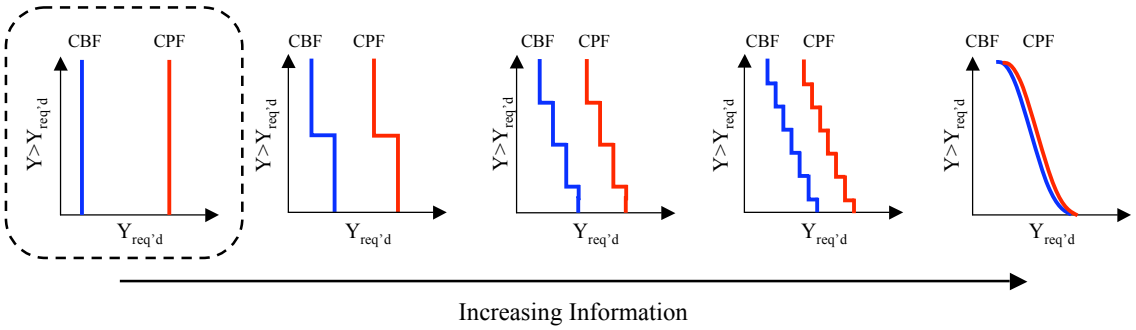


Figure 35: Illustration of Evolving Confidence with Increasing Available Information.

One major issue lies in the evidence theory formulation. The evidence theory implementation proposed for this study is a limiting case. As the amount of information available increases, evidence theory allows for a more specific statement to be made. This is represented Figure 35. The proposed approach for the Fidelity Selection Problem (FSP) is shown with a black dotted box around it. This approach involves specifying simply that the distribution will fall between some upper and lower bound. If more information were available, which is represented in Figure 35, one would be specifying uncertainty divided into bins. The approaches to the right have belief statements made about each bin. If using a frontier finding algorithm, one would have to find the bounds for each possible combination of bins, and it would become exponentially more difficult to evaluate as one moves to the right. However, given that the approach proposed for the FSP uses only the the most basic formulation (black box in Figure 35), it would only require a single evaluation.

The second alternative is to use a space-filling DoE to approximate the full factorial. While this approach may be more expensive than a frontier finding algorithm, it will produce an answer more quickly than the full factorial approach. Additionally, this approach would allow the user to incorporate additional information if it becomes available (moving to the right in Figure 35). An approach using a space-filling DoE to approximate the data from a full factorial would not have difficulties with non-convex problems, as the approach is a relatively "brute force" method and does not use information about the outputs in an effort to intelligently select more samples to evaluate. This approach is illustrated in Figure 34. This approach is very much like that discussed for the full factorial of distributions shapes

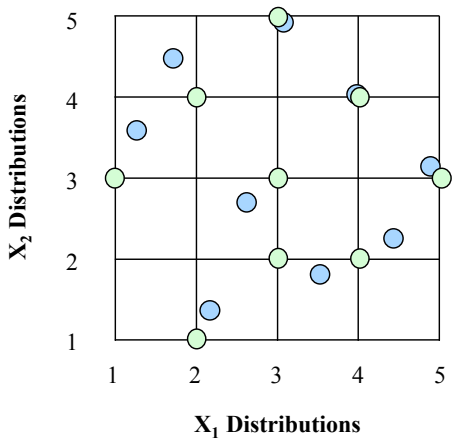


Figure 36: Illustration of Nearest Neighbor Rounding for Space-Filling DoE Samples.

(illustrated in Figure 24) but only some of the cases are being evaluated instead of all of them. The ones that would be run in this scenario are selected using a space-filling DoE.

Using a space-filling DoE approach results in a sampling process much like a nearest neighbor calculation. Here, a space-filling DoE would generate some number of samples, and these samples would then be rounded to the nearest setting corresponding to a distribution. This is illustrated graphically in Figure 36. Here the axes correspond to each variable and there are a variety of different distributions to choose from (distributions 1, 2, 3, and so on). The blue points represent DoE samples. These blue points are then rounded to the nearest neighbor (node) corresponding to a distribution. The rounded points are represented in Figure 36 with green points. These green points would then be evaluated and each is a part of the full factorial previously generated.

One interesting difference between this space-filling DoE approximation and the frontier finding algorithm approximation is how the time cost changes with the amount of information available. As was discussed earlier, this limiting case of evidence theory is the "best-case" for using the frontier finding algorithm because only one series of bins must be evaluated. As more information becomes available the frontier finding algorithm approximation will become more expensive. The opposite is true for the space-filling DoE approximation. In this approximation, the time cost decreases as more information becomes

available. Consider this limiting case. Here one is trying to approximate any possible combination of distribution shapes. If one were instead looking at the right-most scenario in Figure 35, one would have a probabilistic study and only a single distribution combination should be run. Therefore, one could say that this limiting case is the "worst-case" for the space-filling DoE approach and a probabilistic study would be the "best-case".

Based on the trends of the two approximation methods, an additional uncertainty hypothesis and two additional experiments are necessary to identify an alternative capable of achieving the results from the full factorial but doing so in a much more cost effective manner. **It is hypothesized that the space-filling DoE approach will more quickly approximate the bounds obtained from a full factorial than frontier finding algorithms.** If this were the case, then one could completely dismis the frontier-finding algorithm approximation and should maintain a distribution-based approach to the problem. In order to test this a second and a third experiment must be conducted. The second experiment will implement a space-filling DoE to show whether it can approximate the bounds found with a full factorial more quickly, and the third experiment will test whether a frontier finding algorithm can do the same and whether or not it does so more quickly than a space-filling DoE approach.

4.6 Space filling Design of Experiments Approximation

The second experiment was designed to test whether a space-filling DoE could be used to approximate the full factorial results obtained in experiment 1. To test this, the same three test functions were considered: the Schwefel, Michalewicz, and Sphere test functions. Because it was initially unknown how many DoE samples would be required to approximate the full factorial, two options were available. Either DoEs could be regnerated and all points re-evaluated each time additional samples were added or an infinite series DoE could be selected. For computational efficiency, the second option was chosen and a Halton QMC sequence was used. For low to moderate dimensional problems, "banding" is not a concern because there were more samples than the product of the two largest radixes[1]. Two favorable

[1]In Chapter 3, the root cause for Halton QMC "banding" is stated to be that there are insufficient samples to complete the radix. As long as there are more samples taken than the product of the two largest

aspects of the Halton QMC sequence for this problem are that it is a deterministic sequence, meaning any results created are completely repeatable (unlike an LHS DoE), and it is an infinite sequence, meaning that if more samples were needed, the entire DoE would not need to be re-evaluated (unlike an LHS or Hammersley QMC DoE), which offers a significant time savings relative to non-infinite sequences.

Once samples from the Halton QMC DoE were generated, they were rounded to disribution combinations using a nearest neighbor-like calculation described earlier and illustrated with Figure 36. The resulting distributions were then evaluated in the three test functions and the results were tabulated. In order to find the amount of duplicate data, experiments were run which would collect data until 100 % of the bounds achieved in the full factorial experiment were captured.

This experiment can be broken into two different components. The first involves showing whether or not a space-filling DoE can be used to more efficiently select distribution shapes and approximate results previously obtained for full-factorial models. In this part of the experiment, the intent was to define a relative convergence criteria. The second portion of this experiment would then use these rules and apply them to higher-dimensional test functions (more input variables) and identify how much time was required to find the bounds of the space and show how the method scaled with respect to input variables.

Several different parameters were considered in this experiment. In addition to the number of input variables, the number of points per distribution was considered. This parameter was examined at settings of $1,000$ and $10,000$ points per distribution. This range was reduced relative to the previous experiment to simplify the study. A clear trade-off was not found in the previous experiment, so the variable range was reduced. This was done to identify whether a clear benefit could be drawn from either evaluating more different distributions with relatively few points or evaluating relatively few different distributions with many points. As with the previous experiment, the number of different distributions was also considered. This was varied from 10 to 60 different distributions. The number of different distributions was varied in an effort to minimize rounding in the DoE used to

prime bases, "banding" will not be a concern.

select distribution shapes. As was illustrated in Figure 36, the Halton QMC DoE will be rounded to the nearest node which represents a distribution combination. If there are a small number of different distributions, then the distance between nodes is relatively large and the DoE points may be moved a significant amount. It was thought that by increasing the number of different distribution shapes possible, the rounding could be minimized and this may allow the DoE to more quickly identify the bounds of the space.

4.6.1 Truth Data

Truth data for this experiment was obtained through two different methods for the two different parts of the experiment. The first method was for all situations with fewer than 5 input variables (dimensions). Full factorial data for these instances was generated for up to 20 different distribution shapes in the previous experiment. This information was used to calculate the percent relative bounds captured as was done in the previous experiment.

In the second part of the experiment, for situations with 5 or more input variables (dimensions), no full factorial data was available. Instead, relative convergence was assumed to occur at the asymptote, and relative convergence criteria, developed from data in the first part of the experiment, were implemented in the second part of the experiment. In an effort to ensure the asymptote had been reached, the simulation continued for a period of time after the relative convergence criteria was met.

4.6.2 Experiment 2, Part 1 Results

In the first part of experiment 2, space-filling DoEs were used to identify how quickly the bounds for 2 to 4 dimensional test problems could be found. When considering the Schwefel test function, the reader is first directed to Figure 37. Here, the percent relative bounds captured, which is relative to the full factorial data generated in experiment 1, is compared against simulation results for space-filling DoEs for 10 to 60 different distribution shapes and either $1,000$ or $10,000$ points per distribution. As was expected, the results indicated the bounds of the space can be captured relatively quickly using the space-filing DoE approach. The trends seen in results for the Schwefel function for the full factorial are also seen here: the curse of dimensionality remains. As the number of input variables increases from 2

(blue points) to 3 (black points) to 4 (red points), the number of distributions necessary to capture a given percentage of the bounds, regardless of sample size, increases. As one might expect, these trends also confirm earlier results that the number of samples in the distribution increases the amount of information gained on a per-distribution basis. One unexpected trend in the results is with respect to the number of different distributions. Examining the range considered for the study, there is not a strong trend with number of dimensions. While the situations with 30 or 40 different distributions do have a slightly faster convergence than those with 10 or 20, there is not a strong trend. Additionally, there does not seem to be any value added in running more than 40 different distributions.

Moving on to the Michalewicz test function, Figure 38 shows the convergence to 100% of the relative bounds plotted against the number of total distributions evaluated. Here, one should note that while trends similar to those in Figure 37 are seen, it is clearly much slower to converge. This is as expected and is due to the fact that the global maximum and minimum for the Michalewicz function are located in very tall, "peaky" regions of the space. Interestingly, the scenario with 50 different distributions outperforms all others, with respect to convergence as a function of the number of distributions evaluated, though the 30 different distributions case also performs well, and while most scenarios do come close to the actual bounds within the number of simulations taken, only the situation with 50 different distribution shapes actually reaches the bounds for the 4 input variable case. As was the case with the Schwefel function, there is a slight trend with number of distributions, but relative to the trend with number of input variables, it is only secondary.

The Sphere test function results are provided in Figure 39. Each setting of different distributions is able to quickly converge to the actual bounds of the problem, and though they accomplish this, they have slightly different rates. One should notice that even initially, each setting for number of different distributions captures approximately 99% of the relative bounds, which is more than most settings were able to accomplish for the Michalewicz function. This is largely due to the characteristics of the space as discussed earlier.

In order to confirm the belief that the full factorial contains a significant amount of duplicate data, this information was re-examined. Instead of focusing on the number of

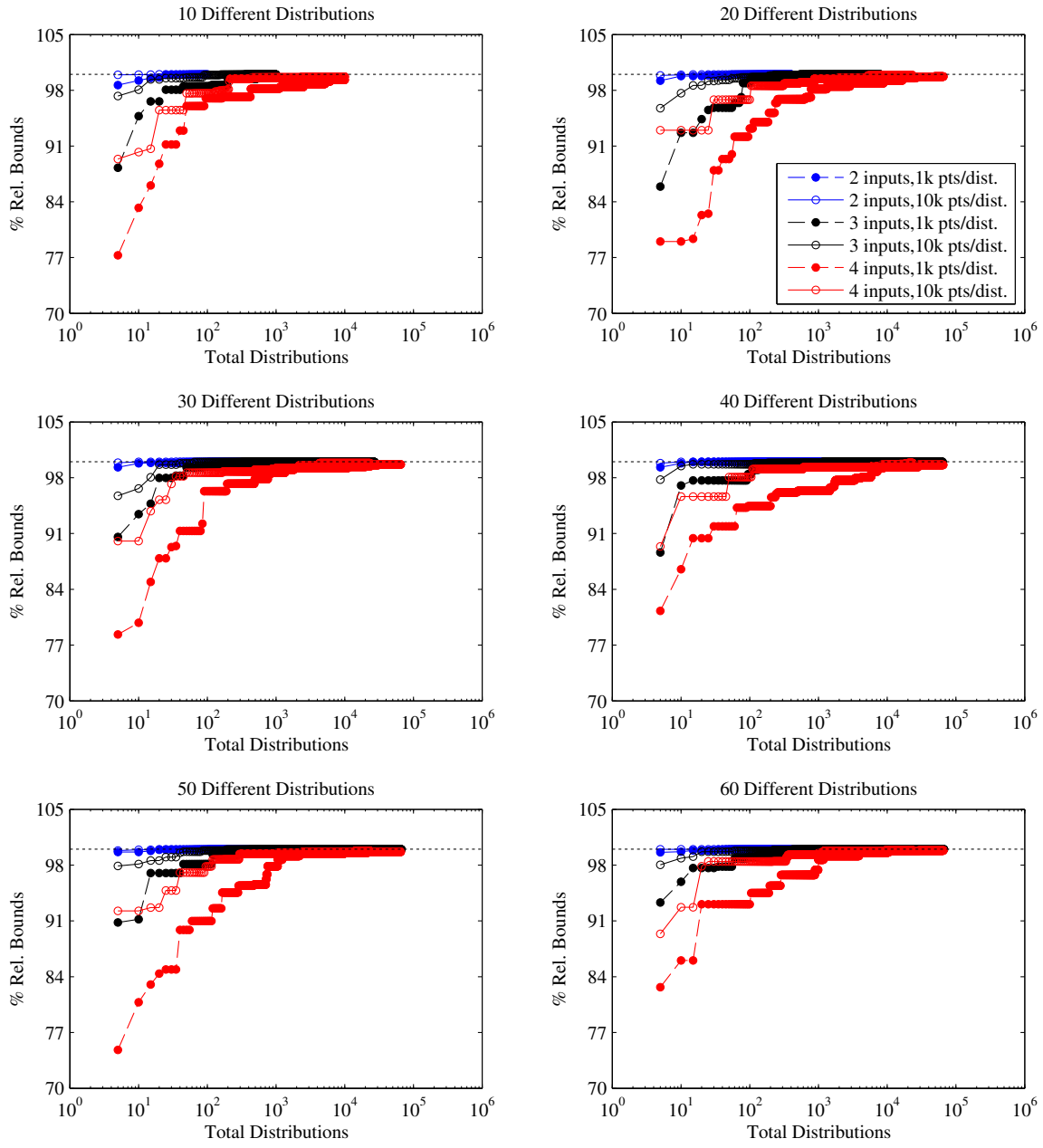


Figure 37: Schwefel Function Percent Relative Bounds Captured vs. Number of Total Distributions Evaluated for a Space-Filling DoE Approximation.

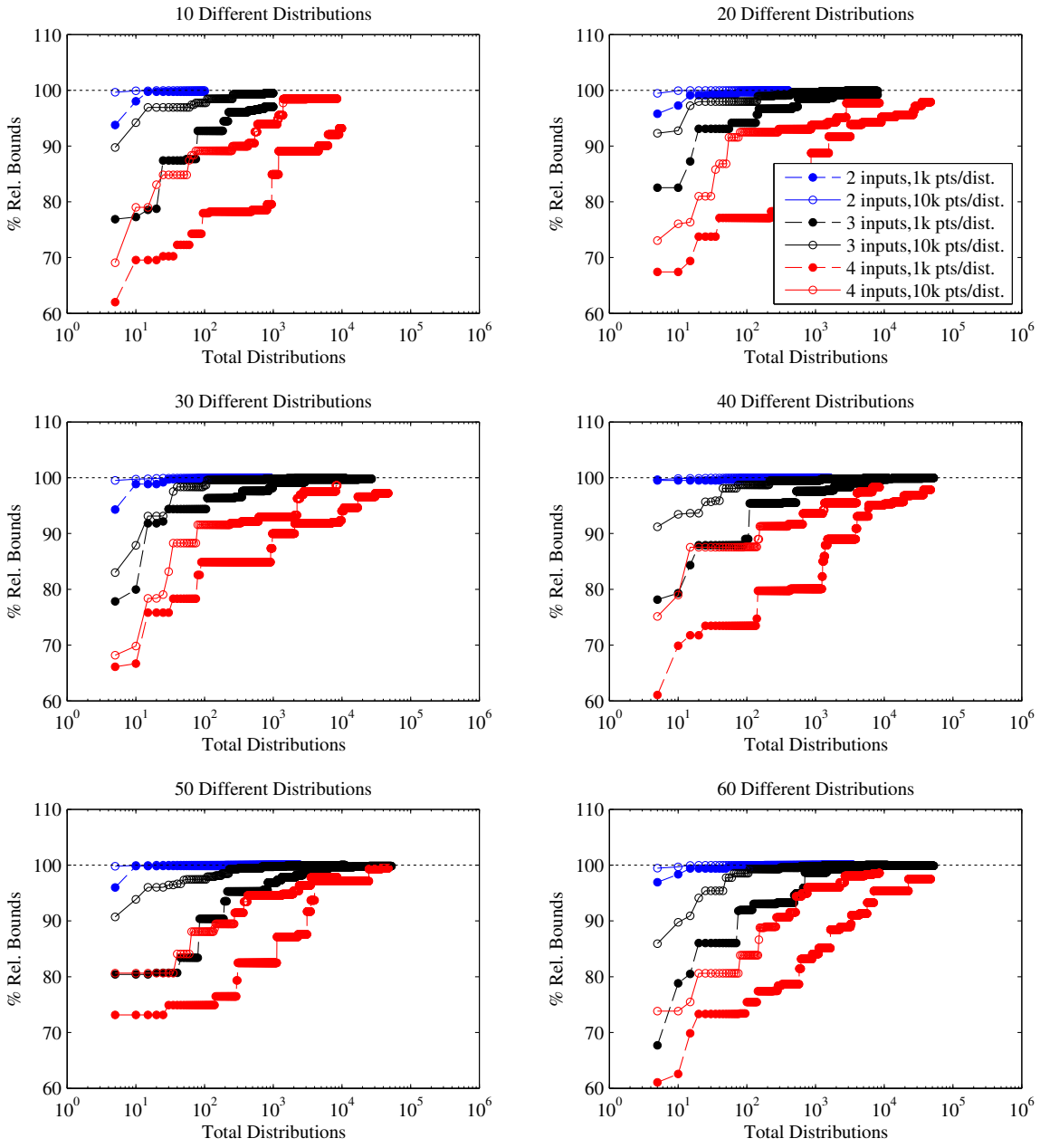


Figure 38: Michalewicz Function Percent Relative Bounds Captured vs. Number of Total Distributions Evaluated for a Space-Filling DoE Approximation.

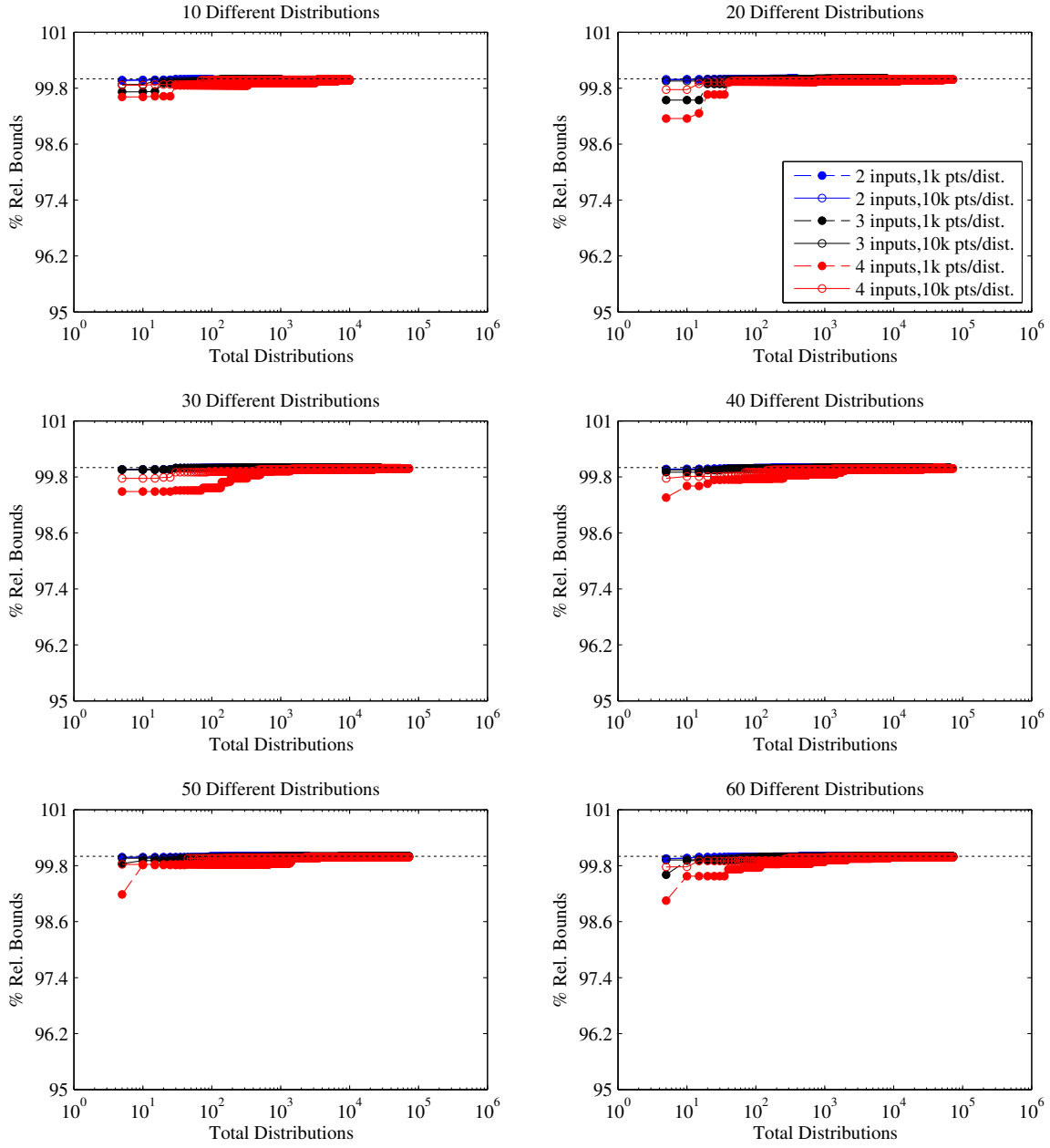


Figure 39: Sphere Function Percent Relative Bounds Captured vs. Number of Total Distributions Evaluated for a Space-Filling DoE Approximation.

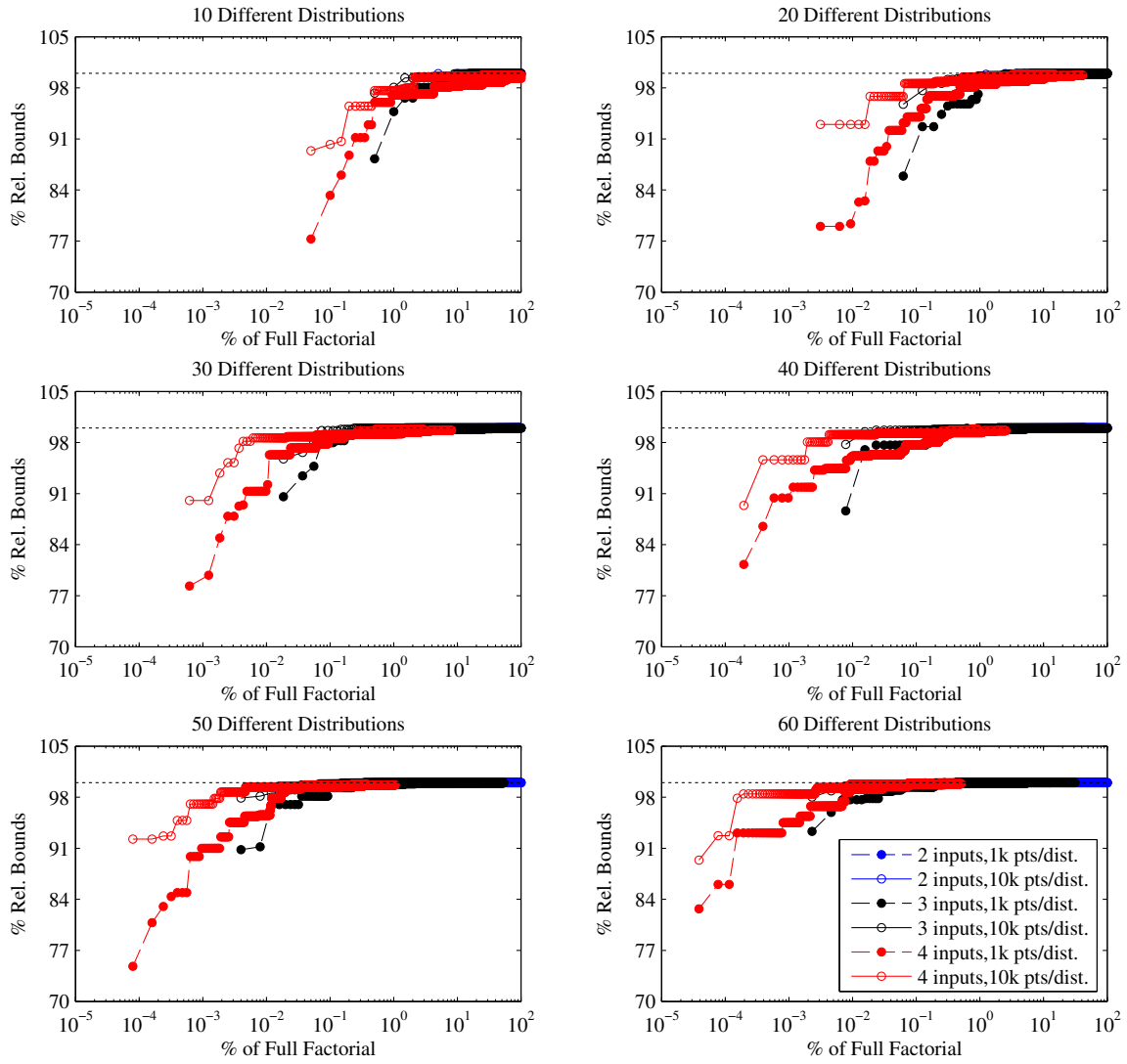


Figure 40: Schwefel Function Percent Relative Bounds Captured vs. Percent of Full Factorial Evaluated for a Space-Filling DoE Approximation.

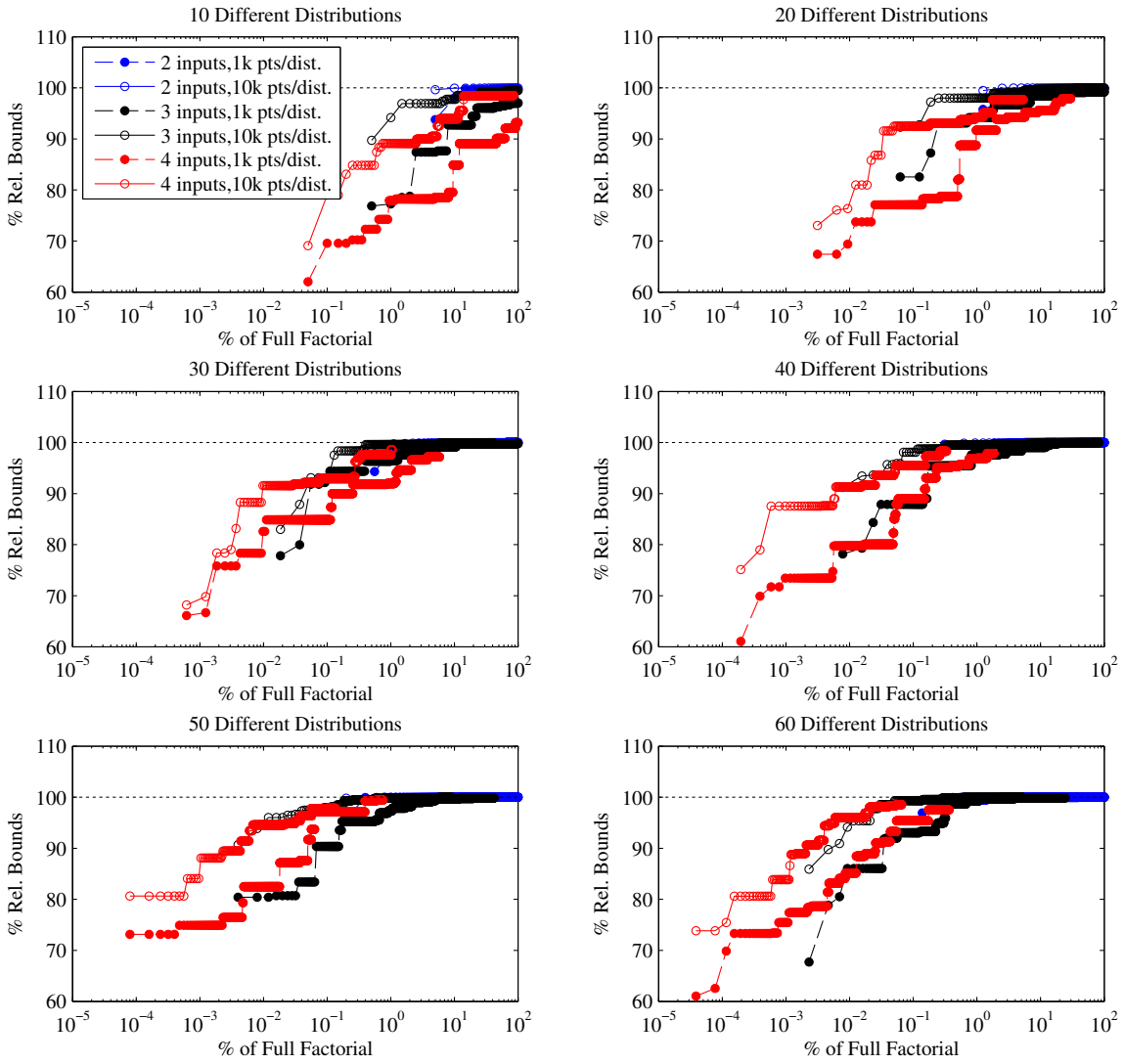


Figure 41: Michalewicz Function Percent Relative Bounds Captured vs. Percent of Full Factorial Evaluated for a Space-Filling DoE Approximation.

total distributions evaluated, one should consider the percentage of the full factorial which is evaluated. Results for the Schwefel function are provided in Figure 40. Here, it is clear that the full factorial contains a *significant* amount of duplicate data. One interesting trend is that while the number of total distributions required to capture the bounds of the space increases with number of input variables, the percentage of the full factorial required to obtain the bounds of the space actually *decreases* with number of dimensions. This trend is illustrated in Figure 40. Here, the bounds for the 2 input variable variant of the Schwefel function converge when evaluating approximately 7% of the full factorial, but this number dramatically decreases to approximately 1% of the design space. A similar trend is seen with respect to the number of distribution shapes evaluated. As the number of different distribution shapes increases from 10 through 60, the percentage of the full factorial also increases. As was shown earlier in Figure 37, there is not a strong trend with number of total distributions; this decrease in percentage of the full factorial is only caused because the full factorial increases with the number of different distributions.

Much like the Schwefel function results, the Michalewicz function results, displayed in Figure 41, show that while the bounds are not necessarily captured in all instances, there is a significant duplication of data and nearly all of the relative bounds identified with the full factorial of distributions can be quickly obtained with a very small portion of the full factorial.

The results for the Sphere test function with percent relative bounds plotted against percent of the full factorial examined is provided in Figure 42. Here, similar trends to those seen for both the Schwefel and Michalewicz test functions are shown. A significant amount of the full factorial contains duplicate information, and in only running a small or possibly even extremely small subset of this full factorial, the bounds of the space can be identified.

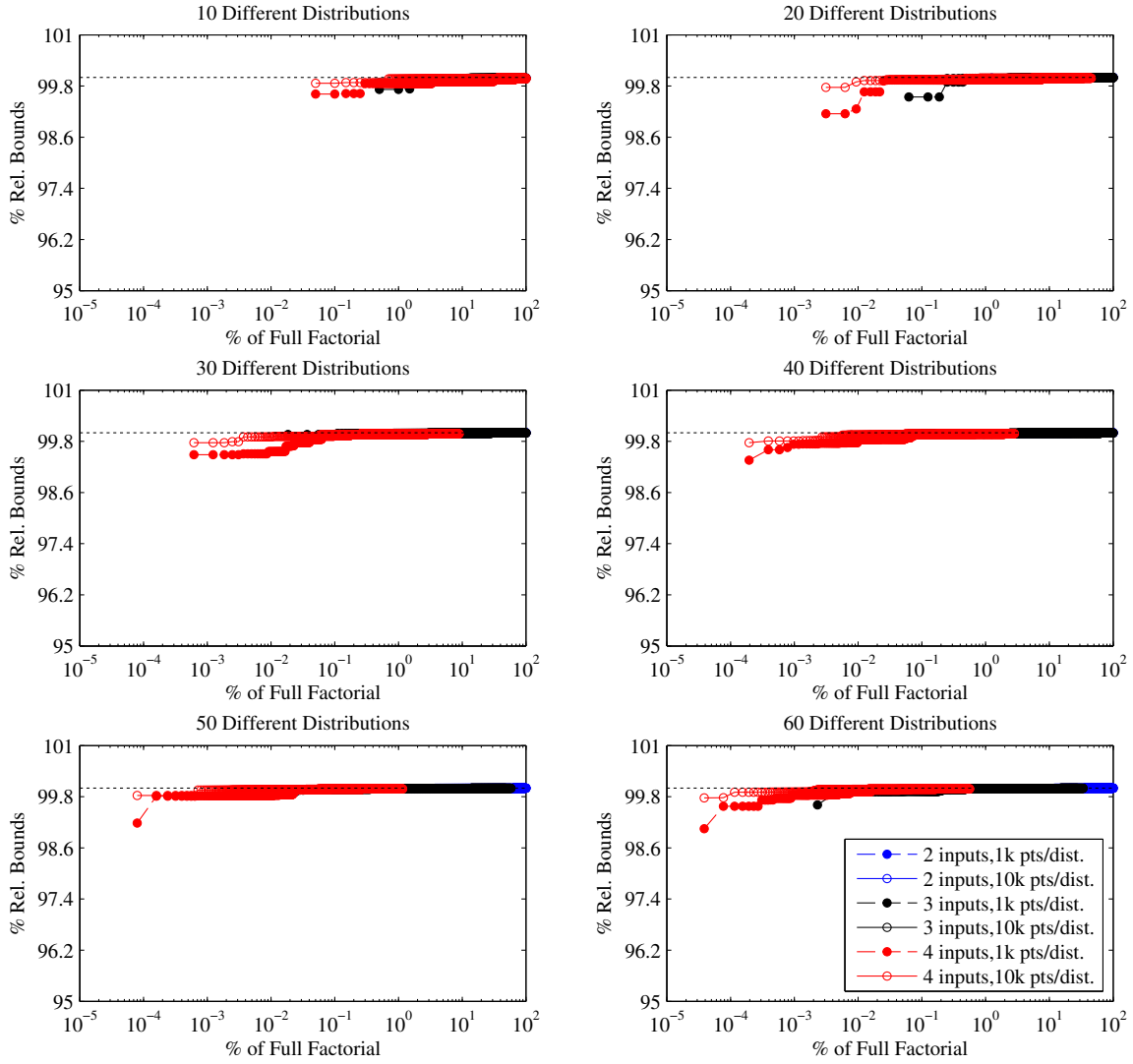


Figure 42: Sphere Function Percent Relative Bounds Captured vs. Percent of Full Factorial Evaluated for a Space-Filling DoE Approximation.

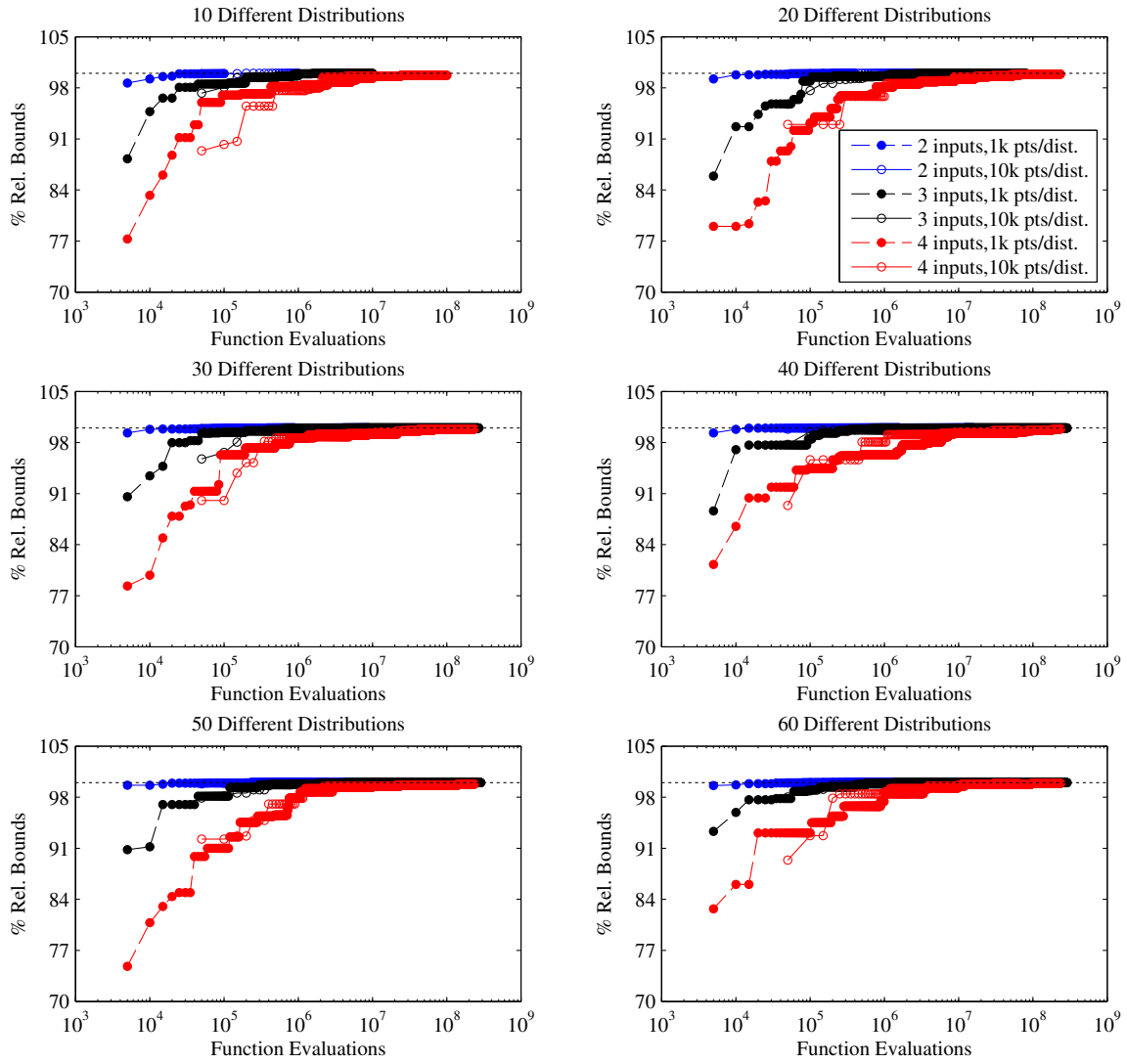


Figure 43: Schwefel Function Percent Relative Bounds Captured vs. Number of Function Evaluations for a Space-Filling DoE Approximation.

In order to evaluate the trade-off between the number of function evaluations and the number of different distributions considered, one should examine the Schwefel test function data provided in Figure 43. In this figure, the number of function evaluations is plotted against the percent relative bound captured. As was the case earlier, there is not a clear difference between choosing to have relative few distributions with many points in each or choosing many distributions with relatively few points in each. One should note that while this is the case, there is value to be gained by having several different shapes of distributions and what is meant by "relatively few" distributions is still approximately 10 to 20 different shapes, as opposed to only 1 shape as would be selected in a maximum entropy approach.

Similar trends were seen for the Michalewicz and Sphere test functions. One should note that while these results show there may not be a significant benefit between using 10, 20, or 30 different distributions and $1,000$ or $10,000$ points per distribution, there is a clear benefit, as shown with the full factorial data, between using 1 and 20 different distributions. Because these results showed that there was no clear "victor" in this trade-off, only distributions with $10,000$ points will be examined from this point on for ease of viewing.

4.6.2.1 Development of Relative Convergence Criteria for Test Functions

After data was obtained, clearly showing that a space-filling DoE could be used to relatively quickly capture the bounds of a low-dimensional problem with only a small fraction of the full factorial and that these bounds were obtained asymptotically, a relative convergence criteria was needed. If a relative convergence criteria was obtained, it could be implemented and allow users to save time and prevent them from needlessly evaluating all cases in the full factorial; the simulation could instead cease once the bounds were reached.

Based on trends found in Figures 37, 38, and 39, there is clearly a trend with respect to the number of input variables examined. Also, while only limited data was available, there was a weak, problem-dependent trend between the number of different distributions and the ability for a method to capture the bounds.

Table 1: Relative Convergence Rules Considered for Space-Filling DoE Approximation.

Rule	Begin Checking After	Stop If No Change After
1	(# inputs) (# dist.)	$\frac{1}{2}$(# inputs) (# dist.)
2	(# inputs) (# dist.)	(# inputs) (# dist.)
3	(# inputs)^2 (# dist.)	(# inputs)^2 (# dist.)
4	(# inputs)^2 (# dist.)	$\frac{1}{2}$(# inputs)^2 (# dist.)
5	(# inputs)^3 (# dist.)	(# inputs)^3 (# dist.)
6	2 (# inputs)^3 (# dist.)	(# inputs)^3 (# dist.)
7	2 (# inputs)^3 (# dist.)	2 (# inputs)^3 (# dist.)
8	(# inputs)^4 (# dist.)	(# inputs)^4 (# dist.)

Data obtained in the first part of this experiment provided deterministic data and allowed for easy testing of possible convergence criteria. Because the number of dimensions was shown to have a strong correlation with the total number of distributions required and the number of different distributions was shown to have a weak correlation with the number of total distributions, these two parameters were identified as terms in the possible candidate rules. The considered rules are provided in Table 1.

Each of these convergence rules would involve the following. After evaluating so many distributions (center column), the algorithm would begin checking how long it had been since the bounds for the space had changed. If they had not changed in a certain number of distribution evaluations (right column), the rule would be completed and the algorithm could stop, having arrived at the asymptote (and true bounds of the space). If the algorithm found that the bounds had changed in the last certain number of distribution evaluations, it would continue evaluating distributions until the bounds remained unchanged. When examining the performance of the various rules, one can easily see that the rules at the bottom of the list are much more stringent than those at the top. These rules were then applied to the data collected for the first four dimensions, and their ability to appropriately terminate was assessed.

When examining each rule's performance on the Schwefel test function, the reader should

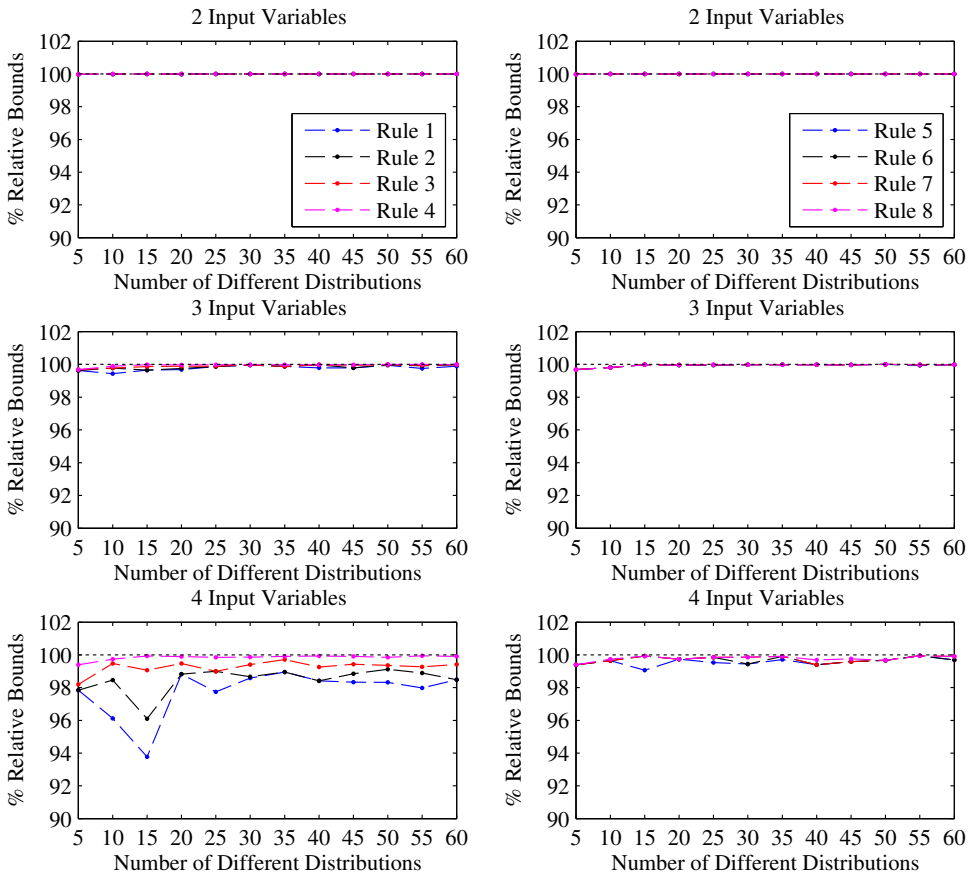


Figure 44: Asymptote Rules Performance for Schwefel Function Space-Filling DoE Approximation vs. Number of Different Distributions.

examine Figure 44. Here, one can see that as the number of input variables increases, nearly all rules' ability to identify the asymptote degrades. The concern with using rules whose performance degrades with these relatively low dimensional problems is that when moving to higher dimensional problems, the performance will not only degrade, but it will do so and the user will not be aware of it. This is a concern which can be seen especially when comparing rules 1 through 4 in Figure 44, but degradation also occurs on some level for rules 5 and 6.

When examining the Michalewicz function, these degradation issues increase considerably. Consider each of the relative convergence rules performance provided in Figure 45. Here not only do issues occur for 4 input variables, but rules 1 through 4 show significant degradation when looking at only 3 input variables. Additionally, when going to 4 input

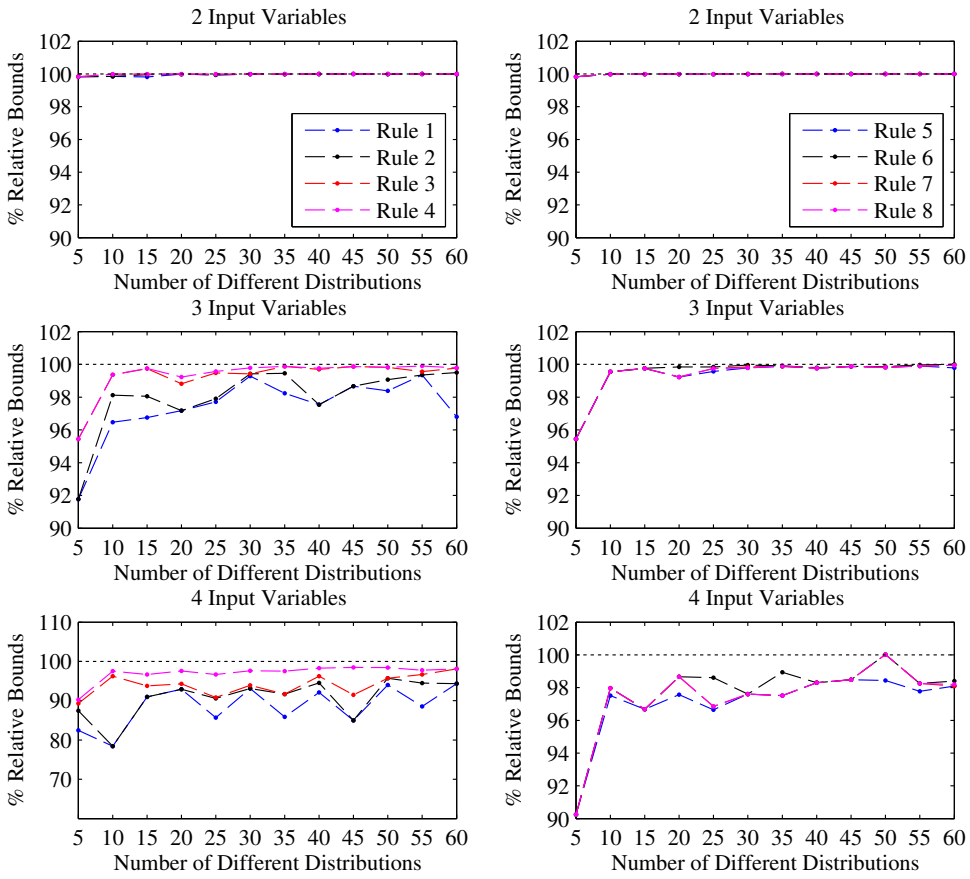


Figure 45: Asymptote Rules Performance for Michalewicz Function Space-Filling DoE Approximation vs. Number of Different Distributions.

variables, all rules show significant issues, and only rules 7 and 8 are able to converge on the actual bounds for 4 input variables; this only happens when 50 different distributions are used.

Finally, the relative convergence rules' performance on the Sphere function is provided in Figure 46. As was seen before and would be expected in each of the rules, even the most lenient rules performed well for all dimensions examined.

Now that all rules have been examined and their performance for all three test functions has been evaluated, a decision should be made. Promising rules should be kept and implemented for higher dimensional problems. Similarly, rules whose performance was poor should be dismissed. Rules 7 and 8 were the only rules which were able to capture the

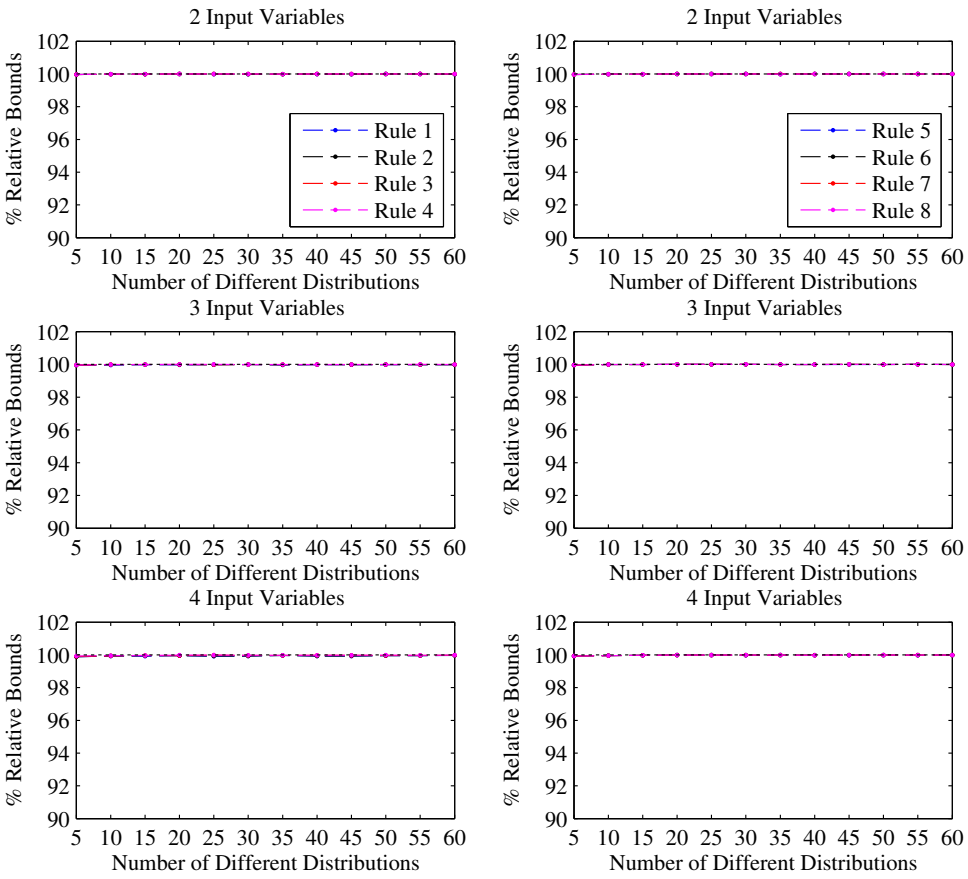


Figure 46: Asymptote Rules Performance for Sphere Function Space-Filling DoE Approximation vs. Number of Different Distributions.

bounds for the Michalewicz function. While nearly all performed well for the Schwefel function, these two rules outperformed all others, especially when considering cases with 35 to 45 different distributions. The Michalewicz function's poor performance should be noted, but the author has not yet seen a "real world" engineering problem with an extreme characteristic such as this. If anything, the Schwefel function should be considered an extreme case for realistic problems. The Sphere function's performance is largely uninteresting as all rules are able to identify the bounds, so any selection would have good performance for this parameter.

One should also consider the amount of time required to conduct the evaluation. The full factorial case required roughly a day to complete for the 4 dimensional problem with 20 different distributions. Completing a 5 dimensional problem with 20 distributions or a 4

dimensional problem with more distributions would be a prohibitively long period of time. This was the motivation for approximating the full factorial results. In shifting from a full factorial approach with 4 input variables to a space-filling DoE approach with rules 7 or 8, the 4 input variable Schwefel and Michalewicz functions' bounds are identified in roughly 3 to 7 minutes for 30 to 40 different distributions. While this is a significant time savings over roughly a day, there is concern that this may not scale well. This concern is reinforced when considering the relatively large exponents which were necessary on the number of input variables in the relative convergence rule. In order to assess their performance, relative convergence rules 7 and 8 were applied to a higher dimensional version of the three test functions, and this was completed in part 2 of experiment 2.

4.6.3 Experiment 2, Part 2 Results

The second part of experiment 2 focuses on applying the relative convergence rules developed for systems where the actual bounds are known, because they were found with the full factorial of possible distributions, to problems where the actual bounds are not known. The three example problems considered for the first part of this experiment where each had between 2 and 4 input variables. Full factorial data could be feasibly generated for these low-dimensional problems, but few "real world" engineering problems have so few variables. In order to show the true utility of the method, its ability to scale up to moderate and high dimensional problems should be assessed. In order to test this, the example problems were examined with more input variables. Situations with 6 and 8 input variables were examined. Based on information obtained for the example problems in the first part of this experiment, situations with 30 to 40 different distributions, $10,000$ samples per distribution, and only rules 7 and 8, as outlined in Table 1, were considered.

Because the full factorial could not be reproduced, two separate stopping criteria were implemented. The first was for simulations to terminate if they met the relative convergence rule. The second criteria was a time out. If the simulation ran for 5 hours and did not converge, it should terminate. This should be taken into account when examining the results.

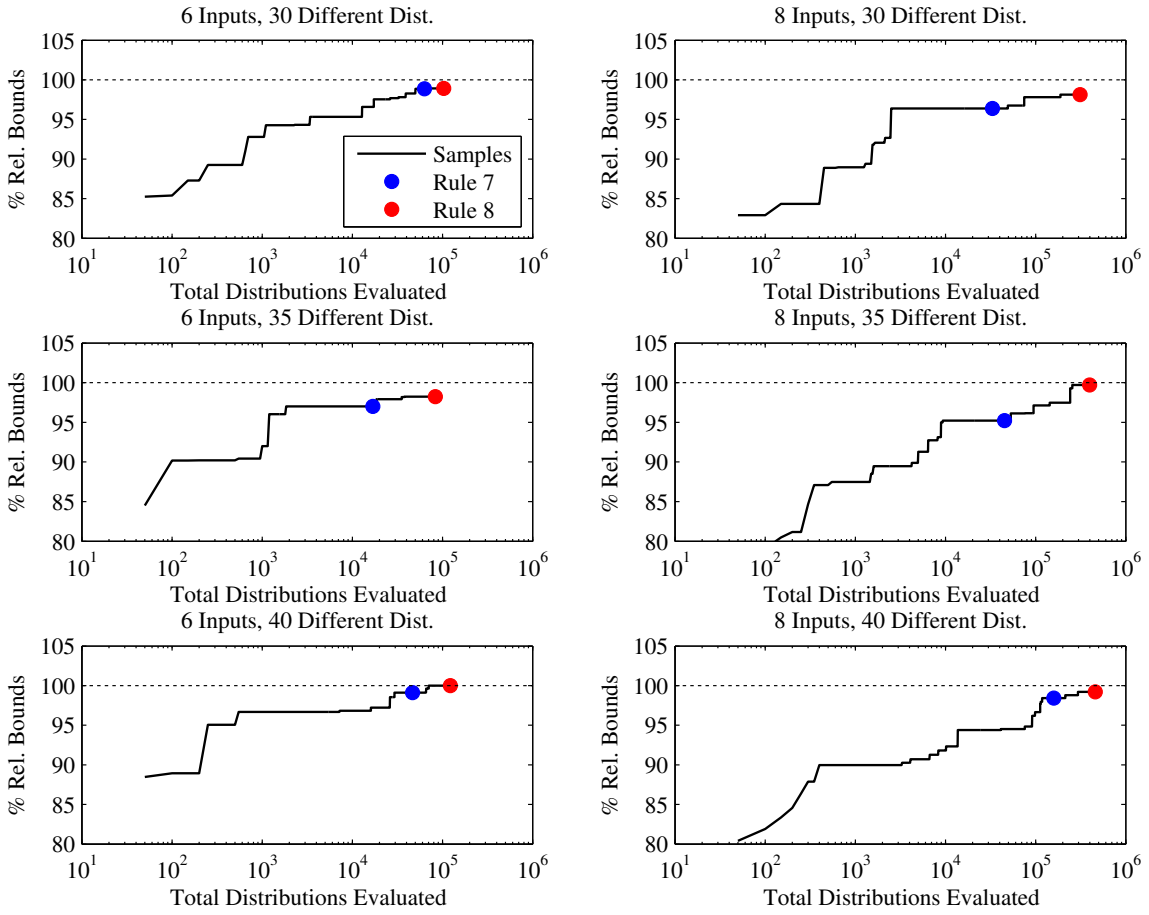


Figure 47: Higher Dimensional Schwefel Function Percent Relative Bounds Captured vs. Number of Total Distribution Evaluations for a Space-Filling DoE Approximation.

Results for the number of total distributions plotted against percent relative bounds captured is provided in Figure 47. In this figure, convergence for rules 7 and 8 are provided as a function of both number of distributions (moving from top row of panes to bottom row of panes) and as a function of the number of input variables (moving from left column to right column). In the first part of this experiment, the maximum bounds found from the full factorial data for a given number of dimensions was used to normalize the data and calculate the percent relative bounds. Because full factorial data was not available for higher dimensional variants of the Schwefel function, the percent relative bounds shown in this figure was instead calculated with the maximum bounds found in this part of the experiment. For this reason, it is interesting to note that for the 6 input version (shown in the left column), only one of the panes is able to capture 100% of the relative bounds, though all seem to be within 3% of the same bounds. A similar finding can be made for the 8 input version (shown in the right column) for the Schwefel function: the relative convergence rules are not converging to the same bounds, but are within 5% of the same value. Given that neither of these rules was able to consistently capture the bounds for this function, it would imply that the relative convergence rules should be more stringent, possibly meaning a larger exponent is necessary for the (# of inputs) term in Table 1. Before any real conclusions can be drawn, more information should be examined.

Moving on to the Michalewicz function, one can view the results in Figure 48. Here, many similarities exist relative to the Schwefel function. Again, the two relative convergence criteria seem unable to consistently reach the bounds. In fact, the differences between the maximum bounds captured for the three different distribution settings is more significant; some distribution settings are terminated as much as 5% from the maximum bounds captured by another setting for the 6 input version. When going to the 8 dimensional version, this is increased further, where solutions are as much as 9% different, depending on the number of different distributions used. These findings reinforce the implication that the relative convergence criteria are not strict enough and are prematurely stopping.

Finally, when considering the Sphere test function, one should recall that for versions with 2 to 4 input variables, all rules examined were able to consistently capture the bounds

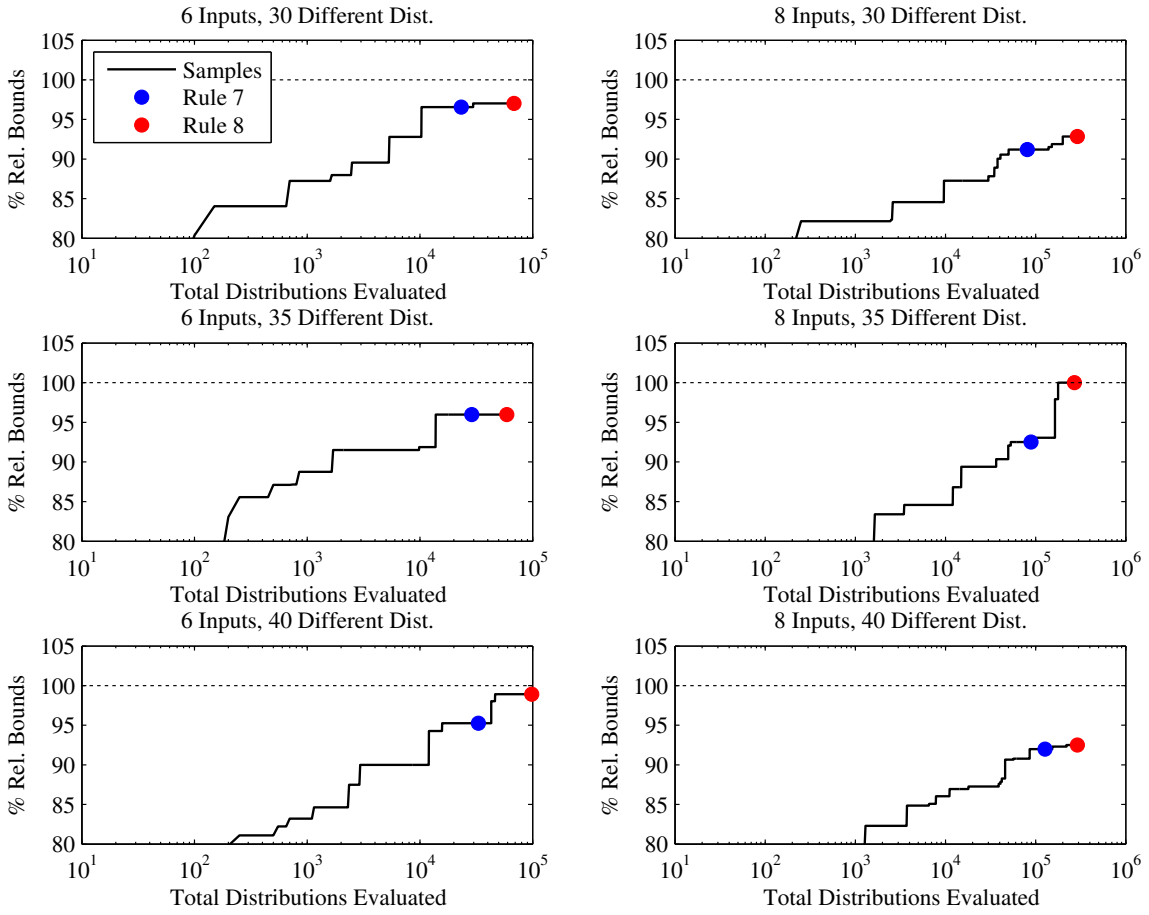


Figure 48: Higher Dimensional Michalewicz Function Percent Relative Bounds Captured vs. Number of Total Distribution Evaluations for a Space-Filling DoE Approximation.

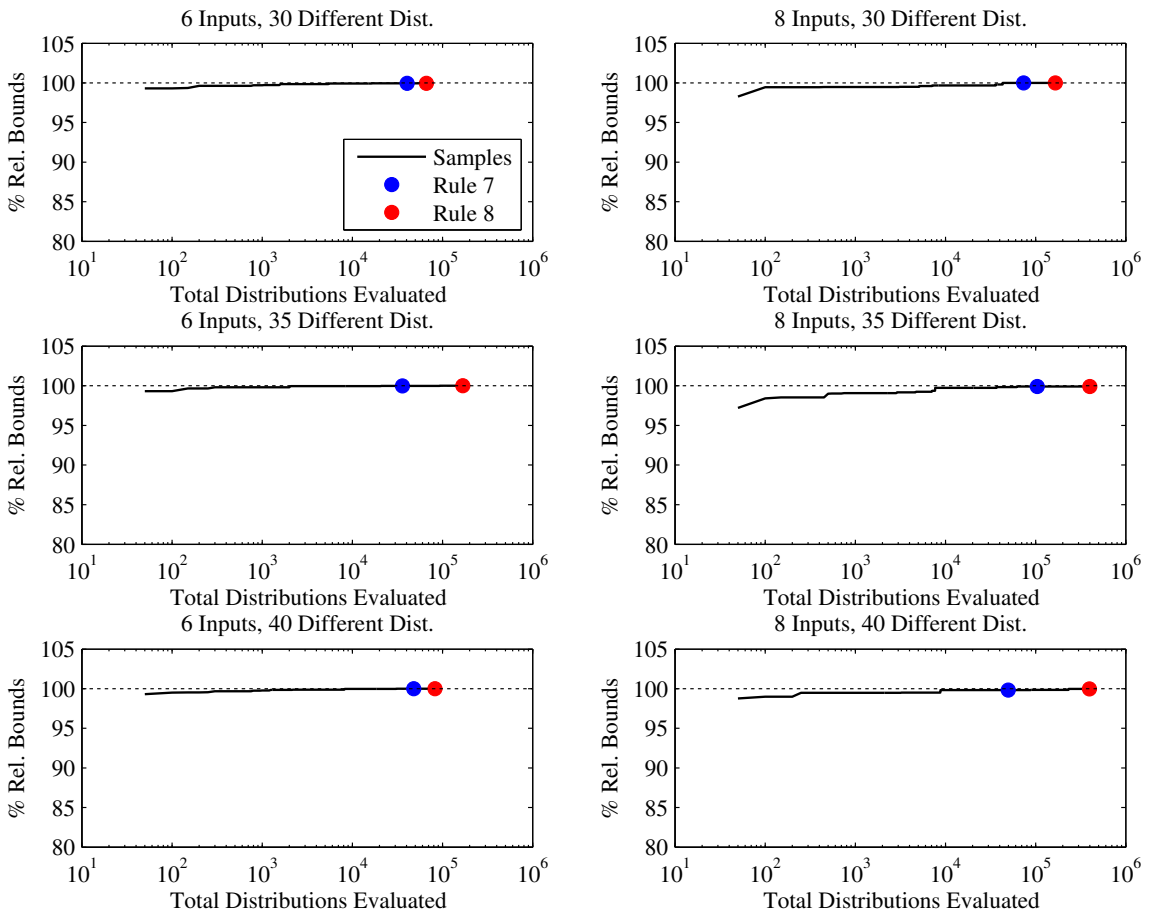


Figure 49: Higher Dimensional Sphere Function Percent Relative Bounds Captured vs. Number of Total Distribution Evaluations for a Space-Filling DoE Approximation.

within a very tight tolerance. The results for the 6 and 8 input version are provided in Figure 49. This figure's findings are much more promising, as the relative convergence rules are able to terminate within 0.1% of the same solution for each distribution setting for the 6 input version and within 0.2% of the same solution for the 8 input version. While these findings are promising, one can see the clear degradation as number of inputs increases, though the magnitude is quite small.

As stated earlier, it was concerning that the relative convergence criteria were terminating prematurely for the Schwefel and Michalewicz functions. If one were to impose a more strict criteria, this would increase the amount of time required to complete the simulation. Before pursuing this enhancement, one should first realize the amount of time spent calculating the bounds found in Figures 47, 48, and 49. The percent relative bounds is plotted

against time (in minutes) for the Schwefel function in Figure 50. Here, one should notice that convergence time varied significantly. Rule 7 was able to find 99.1% of the relative bound in as little as 4 minutes when using 35 different distributions, but when using Rule 8 it requires 37 minutes when using 40 different distributions;, this shows more than a factor of 9 difference in convergence time. A similar variation in time required was seen for the 8 input version. Rule 7 was able to converge with over 98% of the relative bounds in only 11 minutes when using 35 different distributions, while rule 8 required 4.6 *hours* to find 99.2% of the relative bounds when using 40 different distributions. This shows a variation of a factor of 25 in convergence time. While this is not only concerning that there is such a very large variability in the amount of time required, it is also concerning that at best the bounds for the 8 input version can be found in 11 minutes and the time required seems to be growing very quickly with number of inputs.

The amount of time required for the Michalewicz function is provided in Figure 51. Here, the variation in run time is not as significant. Rule 7 converged to roughly 86.5% of the bounds in 15 minutes with 30 different distributions and 6 inputs, but it took over an hour for rule 8 to converge with nearly 99% of the relative bounds when using 40 different distributions. Variation for the 8 input version was slightly smaller, varying by a factor of roughly 4 depending on the setting selections, though this data is notably skewed. As mentioned earlier, a second convergence criteria was implemented, specifically a time-out criteria. The Michalewicz function took over 5 *hours* to converge when considering either 30, 35, or 40 different distributions for the 8 input version with rule 8. Given the significant difference between the Schwefel function convergence times for 6 input variables, one would expect a similarly large factor to exist for the Michalewicz function if it were allowed to continue running for that long.

Finally, when considering the amount of time required for convergence for the Sphere function, trends similar to those for the Schwefel function are again seen when considering Figure 52. Here, all of the methods are able to converge to the same bounds. There is a factor of over 8 when comparing the fastest convergence time (rule 7, 35 different distributions) to the slowest (rule 8 for 35 different distributions) for the 6 input version. It

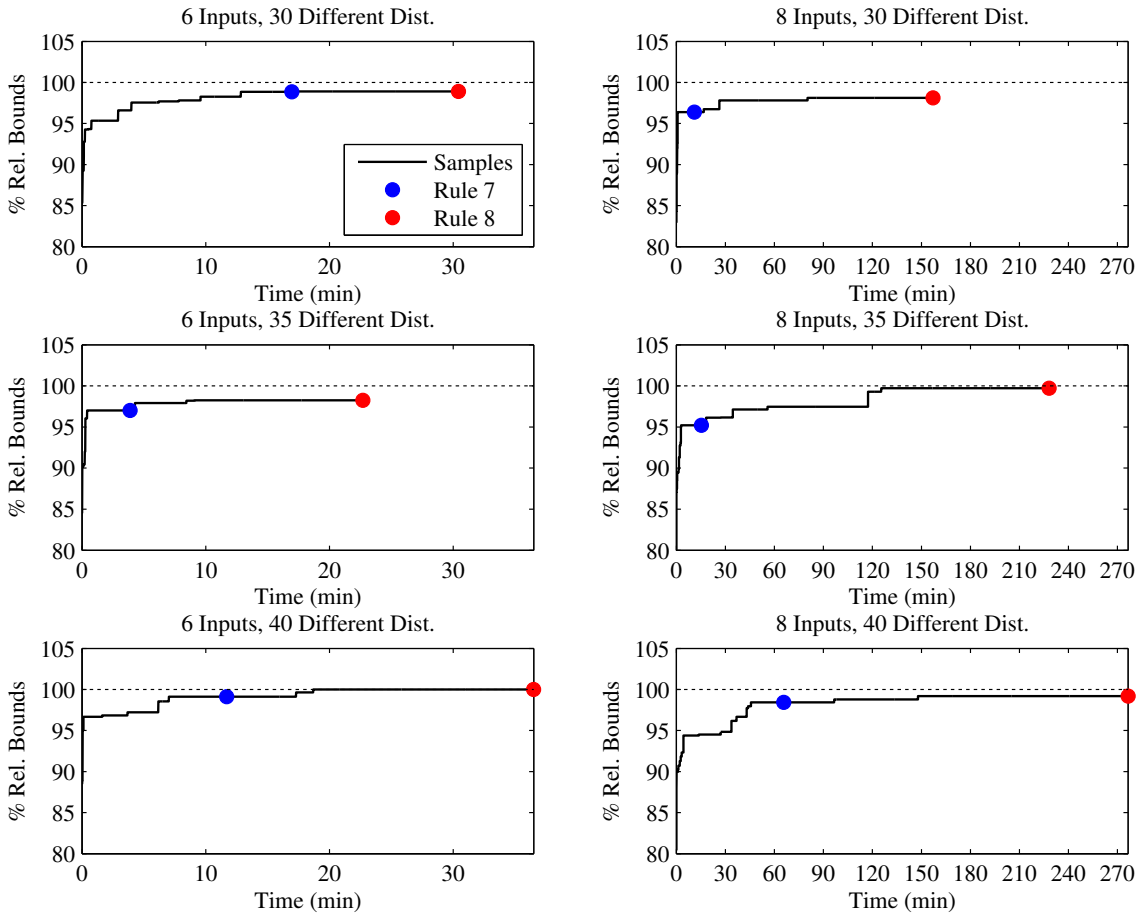


Figure 50: Higher Dimensional Schwefel Function Percent Relative Bounds Captured vs. Time for a Space-Filling DoE Approximation.

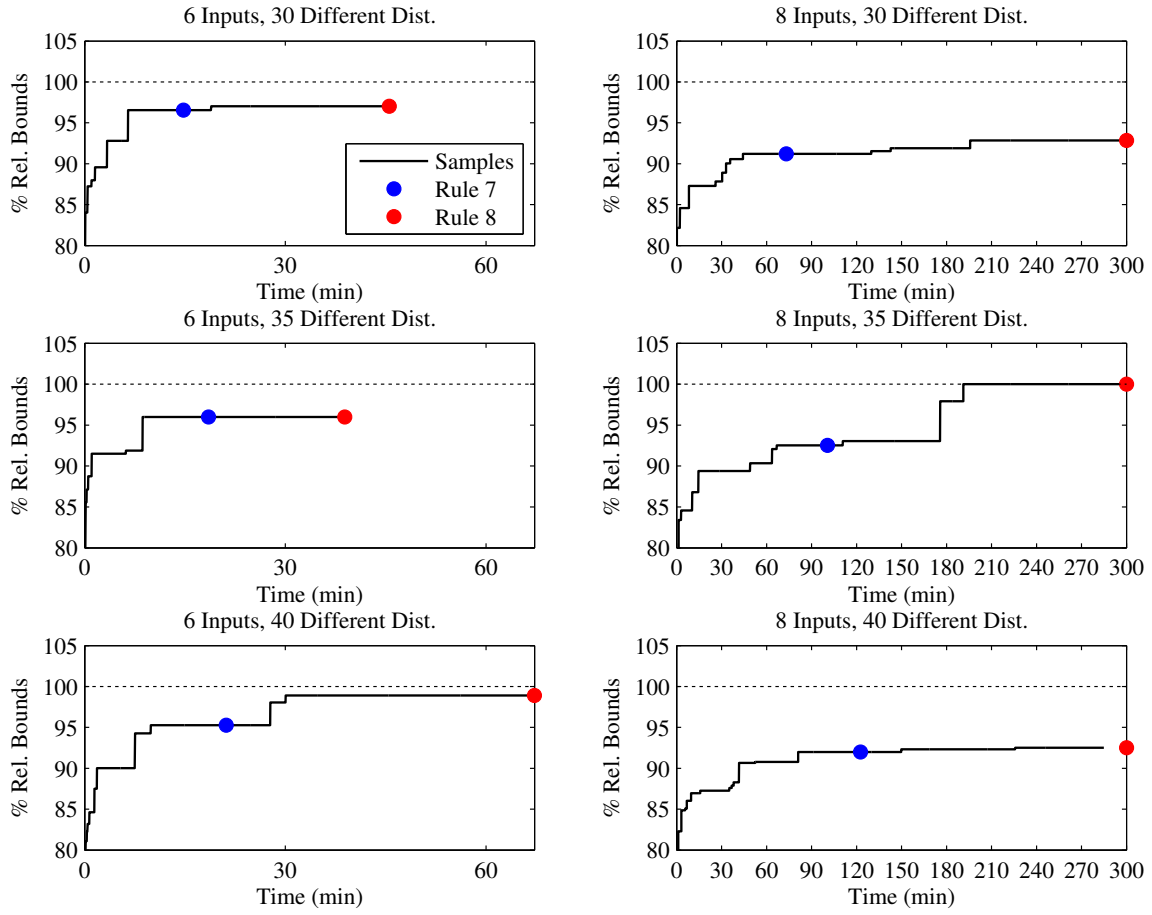


Figure 51: Higher Dimensional Michalewicz Function Percent Relative Bounds Captured vs. Time for a Space-Filling DoE Approximation.

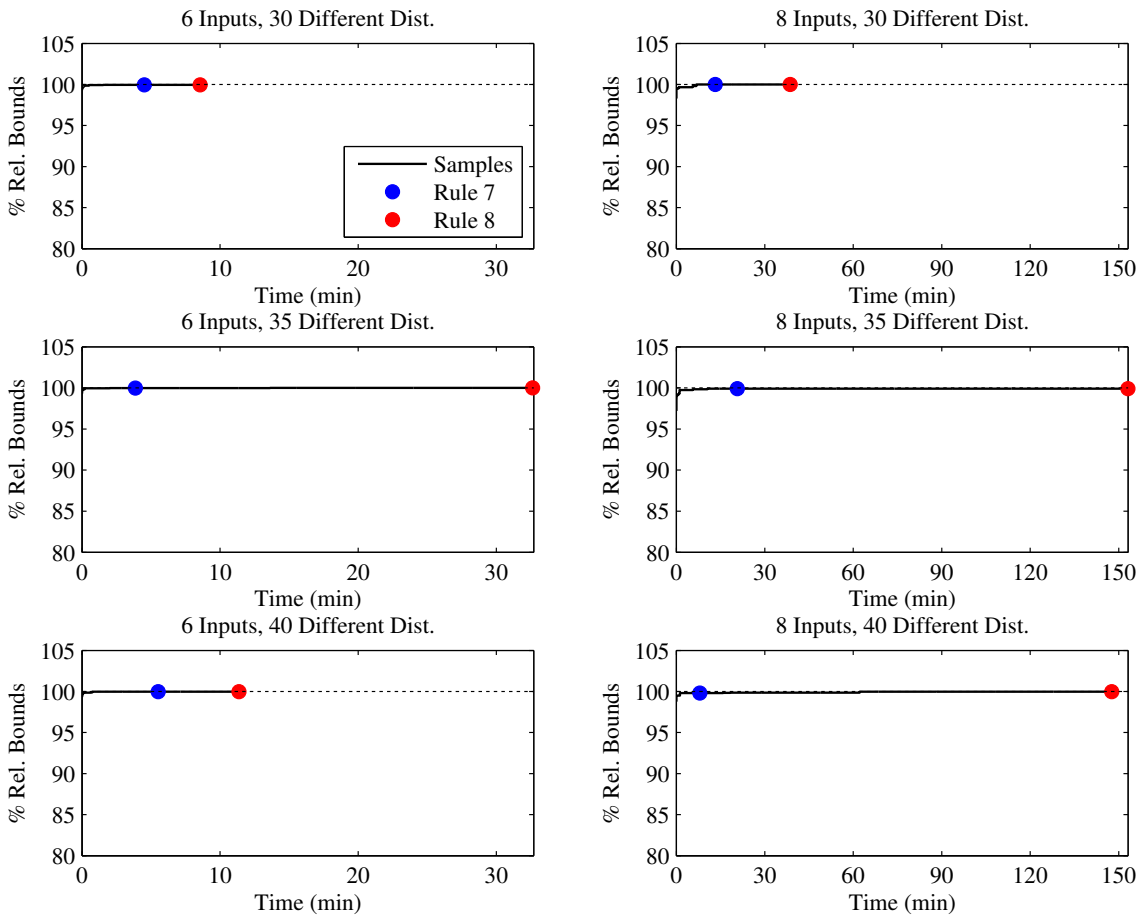


Figure 52: Higher Dimensional Sphere Function Percent Relative Bounds Captured vs. Time for a Space-Filling DoE Approximation.

is interesting that unlike either the Michalewicz or Schwefel functions, the fastest and slowest times are actually for the same number of different distributions. When considering the 8 input version, there is a difference of a factor of over 19 between the fastest and slowest convergence times. It required only 8 minutes to converge with rule 7 for 40 different distributions, but it required nearly 3 hours to converge with rule 8 for the same number of distributions.

The findings from the second part of experiment 2 are somewhat unexpected. A clear relationship between the number of distributions selected and relative performance cannot be established. Depending on which of the promising convergence rules are selected, convergence time was found to differ by up to a factor of 25. While these findings do show that

certain instances will converge relatively quickly, a clear trend cannot be established. Additionally, when considering more challenging problems such as the Michalewicz function, over 5 hours was required to converge with rule 8. Not only was a prohibitive amount of time required to converge the system, the bounds obtained for a varying number of distributions showed a difference in the "solution" of nearly 10% in some instances. This variability in the final "solution" indicates more stringent rules are necessary. When examining the amount of time required, the existing rules already require too much time for most instances. For these reasons, the space-filling DoE approximation should not be pursued for moderate to high dimensional problems. It has been shown to *significantly* reduce completion time over a full factorial combination of different distribution types, but further acceleration is necessary for an evidence theory implementation to be computationally feasible when applying it to the FSP.

A more stringent series of relative convergence rules could be created. These methods would likely require several hours to converge the system, depending on the number of input variables for the system. While this is computationally infeasible for use in solving the FSP, it is not a fundamental issue with using an evidence theory implementation. If the goal of a study is simply to quantify the epistemic uncertainty in a study using an evidence theory approach, this method (though expensive) is not prohibitively expensive. Additionally, it has the benefit of being able to quantify epistemic uncertainty using evidence theory as the amount of available information increases. For this reason, this novel sampling method is recommended when the goal of a study is simply to quantify the epistemic uncertainty in a moderate dimensional problem.

4.7 Conclusions

In this chapter, experiments were conducted to test how one should create a variety of distributions to quantify epistemic model uncertainty given an evidence theory formulation. The results showed that the bounds of the space could be found by assessing all possible combinations of distributions. However, this approach does not scale well, and when looking at moderate dimensional problems, this approach is computationally infeasible. Two

approximations were considered: that of using a space-filling DoE to strategically sample the full factorial and that of using frontier finding metaheuristic algorithms. The first approximation alternative was implemented in this chapter. Results showed that this method could successfully approximate a full factorial while only taking a small percentage of the samples in a full factorial. While the space-filling DoE approach does scale better than the full factorial, it also becomes computationally infeasible when considering moderate dimensional problems. For this reason, the second alternative, using metaheuristic frontier finding algorithms to capture the bounds of the space, should be explored. A review of relevant literature in the field of metaheuristic algorithms is provided in the next chapter. Following that, their implementation and performance when approximating the full factorial will be examined.

CHAPTER V

REVIEW OF RELEVANT LITERATURE ON METAHEURISTIC OPTIMIZATION ALGORITHMS

5.1 Introduction

This chapter will survey relevant literature related to metaheuristic optimization algorithms. These consist of optimization algorithms commonly used to quickly solve Nondeterministic Polynomial (NP)-hard problems. A common mistake is that NP actually stands for "not polynomial".[171] While these problems are indeed not polynomial, it actually refers to the nondeterministic methods commonly used to solve them.[41] If a problem is NP-hard it means that if an algorithm makes the best local policy decision it is not guaranteed the algorithm will reach the globally best result. Policy decisions can be likened to an optimization algorithm's gradient or probe step for continuous problems or a choice in a discrete problem. In optimization terms, an NP-hard problem is a multi-modal problem. NP-hard problems differ from Polynomial (P) problems because, given any starting point, one can reach the best solution to a P problem by always making the best local policy decision. There are a wide variety of P, or unimodal, optimization algorithms, though they will likely not be of use in finding the boundary of a space. Methods used to solve NP-hard algorithms avoid being trapped in local minima by using populations of samples and nondeterministic policy transitions. Given significant recent development in the field of metaheuristic algorithms, it is believed that given the appropriate objective function, these algorithms can successfully find the boundaries of the space.

Unlike deterministic, unimodal optimization algorithms, metaheuristic algorithms include upper-level guidelines or "master strategies" which guide lower-level heuristics to achieve goals.[68] While this may seem like an odd approach, it has proven to be a quite successful class of algorithms for certain approaches.

Three main algorithms will be discussed: Genetic Algorithms (GAs), Ant Colony Optimizations

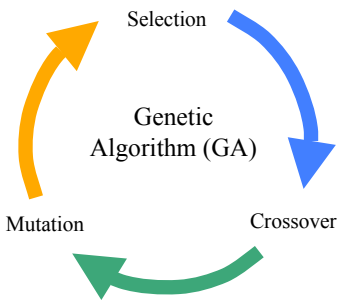


Figure 53: GA Phases for Each Generation.

(ACOs), and Particle Swarm Optimizations (PSOs). After discussing each of these methods, alternative methods will be introduced and their similarity to each of these methods will be discussed to show how assessing the performance of these three algorithms will allow one to roughly assess the performance of their derivatives.

Finding the frontier of a space will require continuous, multi-objective optimization. The input variables for uncertainty quantification are the upper and lower bound in the "adder" functions previously discussed in Chapter 2. However, many of these algorithms were developed for discrete, single objective problems. For this reason, the following outline will be followed to survey these methods. First, the original method will be introduced. Next, notable improvements including allowing them to evaluate multi-objective problems, continuous problems, and constrained problems will be discussed. Each section will conclude with a discussion of how these algorithms can be implemented for this study.

5.2 Genetic Algorithm

GAs are a metaheuristic optimization method developed by Holland, a professor at the University of Michigan in the 1960s.[79, 118] This algorithm was intended to mimic Darwin's view of evolution.[212] Much like with genetic material (chromosomes), a binary string is created to represent each concept. In this algorithm, a population of possible solutions is evaluated, and much like biological populations or organisms, they go through three phases in a given generation. These phases are illustrated in Figure 53.

In the first phase, a population will in essence "battle" to determine which are the most favorable members who should be allowed to breed and have their children move on to

the following generation. Each member's fitness is an evaluation of the objective function for optimization. GAs always assume the fitness should be maximized, so if the objective is to minimize, it is typically multiplied by -1 or the inverse is taken. Popular schemes for selection include tournament selection (points are randomly selected, their fitness is evaluated and the member with the highest fitness moves on to the next generation) and proportional selection (each member's genetic material moves on to the next generation proportional to its fitness relative to the other population members).

The second phase of a GA is the crossover of genetic material. This consists of producing two "children" based on some combination of the "parent" (post-selection) binary string. Here, several schemes are commonly used for crossover. These range from single-point crossover (swapping a single bit in each "parent" to make two new "children") to two-point crossover (identifying a starting and ending bit and swapping all material between these bits in the "parents" to produce "children"), among others.

The third phase of a GA is mutation. This, much like biological evolution, is a random variation. Various mutation schemes exist, ranging from assigning a probability to randomly selecting a single bit in a "child" binary string and flipping it to assigning a much lower probability and using this to determine whether each bit in each child should be flipped.

These three phases are repeated until some relative convergence criteria is met. Specific rules do not exist, and are commonly very problem specific. If a GA were to be implemented to capture the bounds of a space, a relative convergence criteria must be developed to terminate after a period of time or number of generations. This would likely be a function of the number of inputs and/or outputs.

Because of its biological analogy, this type of formulation has gained considerable popularity and has been used to address a variety of different applications, ranging from supersonic business jet design[26] to the design of hybrid rocket motors[199].

5.2.1 Nondominated Sorting Genetic Algorithm - II

Nondominated Sorting Genetic Algorithm - II (NSGA-II) is a popular method published by Deb [50] to identify a frontier of nondominated solutions. In his 2002 paper, Deb outlines

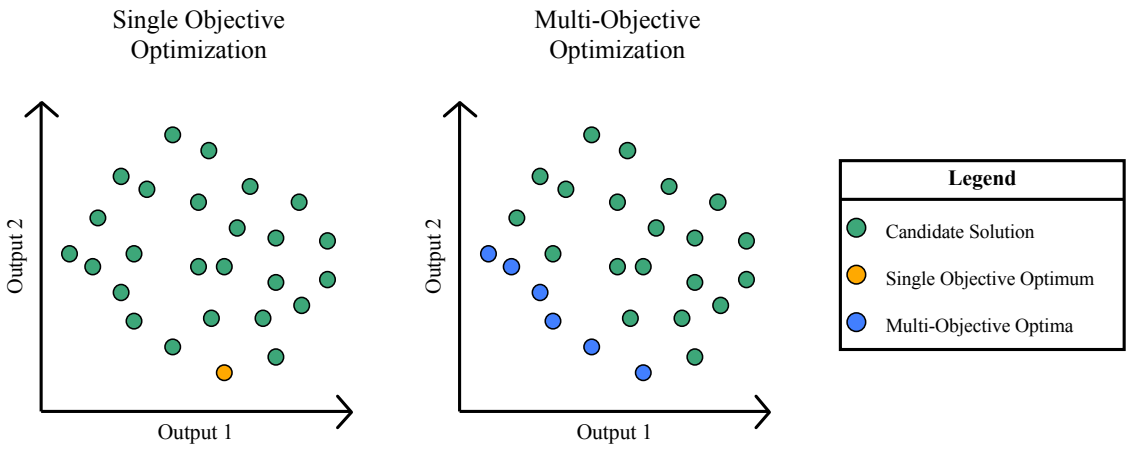


Figure 54: Illustration of Difference Between Single and Muli-Objective Optimization.

his chief modification to the algorithm, which is an alternative method for performing the selection phase of a GA. In a simple GA, as discussed earlier, a single objective is maximized or minimized. Deb's modification instead identifies the front of nondominated solutions to a series of objective functions. A candidate solution is defined to be nondominated if improvement cannot be gained in a given dimension without accepting a degradation in another dimension. An illustration of the difference between a single objective minimum and a frontier of nondominated solutions is provided in Figure 54. In this figure, the left pane provides a series of candidate points; if the objective were to minimize Y_2, the orange point would be the optimum. Similarly, if one were instead interested in identifying all of the "best" points when minimizing both Y_1 and Y_2, the right pane would be of interest. Here, all of the "best points" represent a front of nondominated solutions and are shown in green. Deb's NSGA-II works to identify the nondominated solutions in a population and uses their rank (or which front they fall on) as a fitness evaluation.

A second improvement Deb offers in NSGA-II is the idea of an elitist population. In the simple GA, a population will undergo selection, crossover, and mutation. At the conclusion of the generation, all parent solutions are discarded and the children are kept. Elitism, in the sense employed in NSGA-II, will compare the n parents to the n children ($2n$ solutions) and then the n most fit points will move on to the next generation. If an entire front cannot move onto the next generation, a crowding distance on the front is calculated. This

Listing 5.1: Crowding Distance Calculation.

```
1  function[distance] = crowdingDistanceCalculation(Y)
2  [pop,numY]=size(Y)
3  distance=zeros(pop,1)
4  for each numY
5      sort [Y distance] by column numY
6      distance(1)=Inf
7      distance(pop)=Inf
8      denom=Y(pop,numY)-Y(1,numY)
9      for i=2 to numY-1
10         distance(i)=distance(i) + ( Y(i+1,numY)-Y(i-1,numY) )/ denom
11     end
12 end
```

is algorithmically described in Listing 5.1.

In this calculation, the solutions on a given front are sorted (line 5). The extreme values are then always kept (lines 6,7) and a length calculation is performed to determine how much space on a given front a specific point occupies. Points with a larger distance at the result of this calculation make up a larger portion of the front, and are therefore more important. This calculation will help maintain and promote diversity on a given front of solutions.

5.2.2 Real-Coded Genetic Algorithm

Real-Coded Genetic Algorithms (RCGAs) are another adaptation of a typical GA (or perhaps more appropriately, a Binary-Coded Genetic Algorithm (BCGA)). RCGA was developed in an effort to address problems due to the Hamming distance in BCGAs. The Hamming distance is defined to be the number of bit flips required to move from one position to another. As bit strings become increasingly longer and solutions approach an optimum, Hamming cliffs form, meaning a huge number of bits must be simultaneously flipped for a solution to move one position to the right or left (consider moving from 1000000 to 0111111 or back).[49] BCGAs have difficulty addressing hamming cliffs because a typical binary coding may only require a single bit flip, depending on the point in the sequence, but may require as many as nearly all bits to be flipped.

Gray coding has been used as a method to mitigate this issue. Gray coding is an

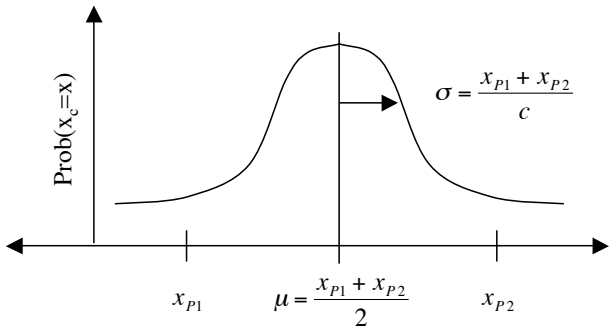


Figure 55: An Illustration of a Blending Process for RCGA Crossover.

alternative binary coding which allows only a single bit to be flipped to move a point one position to the right or left. While this is an improvement, the location of the "one bit" moves throughout the sequence and therefore to truly address the hamming distance issue, a mutation or crossover procedure would have to identify this specific bit.

While Gray coding may make it easier move one position to the left or right, it is still difficult to do. This issue gives rise to the popularity of RCGAs. RCGAs allow for one to more easily move to the right or left, but not without some issue. Crossover, an operation well suited to binary strings, becomes a more complex procedure when addressing continuous variables. A variety of schemes have been identified in the literature, ranging from simplistic averaging schemes (where for a given variable, the child is the average of the parents' value) or blending schemes such as that outlined in Figure 55, among others.[49] In Figure 55, a normal distribution, where the mean is the average of the two parents' value and the standard deviation is some ratio of the parents' average value, is constructed and randomly sampled.

While crossover may be more difficult to appropriately represent in an RCGA, mutation is much simpler. Here, a common approach is to randomly sample from a normal distribution that has a mean equal to the child's value for that variable and a standard deviation that is some constant.

RCGAs have been shown to more quickly converge on an optimum solution than their binary-coded counterpart; additionally, operators such as mutation and crossover are shown to help refine a solution rather than flip bits, potentially leading to a catastrophic result.[79]

RCGAs should be used instead of BCGAs when examining continuous problems, which include frontier-finding scenarios.

5.2.3 Genetic Algorithm Implementation for Frontier Finding

When identifying the frontier of a continuous space, aspects of both Deb's NSGA-II and RCGA should be used. The resulting hybrid, called an Frontier Finding Real-Coded Genetic Algorithm (FFRCGA) was created. The GA is real-coded because continuous variables are being used and the literature shows that a real-coded variant will outperform a binary-coded one. It should be noted that this was not simply accepted. A trade study was conducted by the author, and when developing a frontier finding GA, both binary- and real-coded versions were developed. The findings confirmed what was found in the literature the real-coded version outperformed the binary-coded one. An RCGA will smoothly approach the optimum of a problem whereas a BCGA will chaotically "jump" around the space and eventually reach the optimum.[118]

In order to identify the frontiers of the output space, the nondominated aspect of NSGA-II was implemented with some modification. Rather than searching for a single, desired front of nondominated solutions, all fronts of nondominated solutions should be examined. This requires calculating 2^y different frontiers, where y is the number of output variables examined, instead of the one front identified in Deb's NSGA-II. Because multiple frontiers were of interest, additional modification was needed. The algorithm was initially implemented where all frontiers were examined by the same population. These results were less than satisfying and some further modification was necessary. Deb's NSGA-II sorts all population members first by rank in ascending order and then by crowding distance in descending order. This implementation for multiple fronts shows bias towards some frontiers over others. For this reason, a surrogate for the crowding distance was used. The priority of a solution on a given front was determined. This is much like the crowding distance, but instead this is simply the index for the crowding distance vector for a given front. When sorting using this method, all solutions with a priority of 1 and 2 will move on. These represent the corners for each front. Then, solutions with the highest crowding distance on a given front will

be selected. This implementation means that the same number of points from each front will move on to the next generation. While this improved the performance of the Frontier Finding Real-Coded Genetic Algorithm (FFRCGA), it was still unsatisfactory.

When only trying to calculate all fronts with a single population, an interesting phenomena occurred. If a given front was easier to calculate than another, points would more quickly appear in lower ranks on that front. The "rank" is determined by the nondominated sorting and is determined the crowding distance calculation. The two extreme points on a given front are ranks 1 and 2 whereas all other points are ranked based on their relative crowding distance. This would, in essence, take a solution on one front (rank 6 on front 2) and replace it in the elitist population with a different solution (rank 4 on front 3), because this solution had a lower rank. While the priority calculation would improve diversity relative to a simple crowding distance calculation, this does not account for the possibility of a point on a given front being replaced by another on a different front. In order to address this, a "segregationist" adaptation was proposed. For a sample population of n solutions, several subdivisions would be made. The n solutions would be divided by 2^{ny} different groups (each with $\frac{n}{2^{ny}}$ solutions) and each group would search for a different approach. This method was found to effectively find each front and do so while assigning an equal number of points to each front. This "segregationist", "elistist" FFRCGA was found to have good convergence properties relative to a non-segregationist BCGA on all fronts and was implemented for this study.

5.3 Ant Colony Optimization

ACO is an algorithm developed by Dorigo while working on his Ph.D. dissertation in the early 1990's that was developed to solve Traveling Salesman Problem (TSP).[60] While Dorigo's original method was found to perform more slowly than competing methods, it was significantly improved soon thereafter by both Dorigo and other academics.[24] Dorigo's ACO, originally called Ant System, is a metaheuristic optimization method which was created in an effort to mimic the behavior of Argentine ants foraging for food.[24] The algorithm, much like a GA, is a generation-based algorithm and a cyclic illustration is

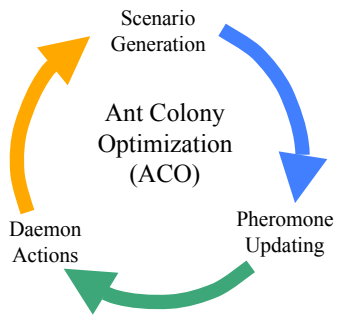


Figure 56: ACO Phases for Each Generation.

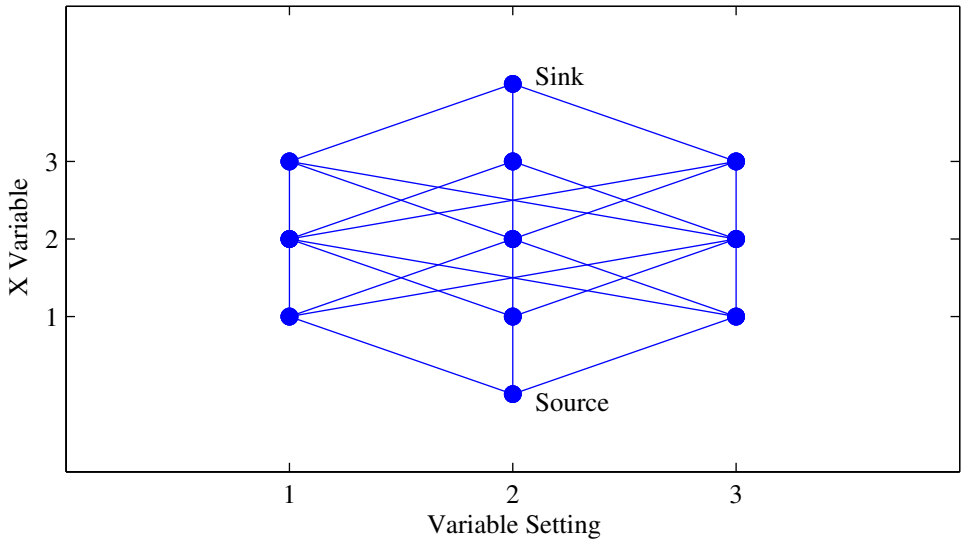


Figure 57: Graph Representation of Variable Discretization for Implementation in ACO.

provided in Figure 56.

The first phase, scenario generation, involves creating a population of virtual ants and allowing them to propagate through the graph. When initially formulated, each node would represent a city. For solving this problem, a graph provided in Figure 57. In this figure, the nodes are shown with blue dots and the edges are shown with blue lines. All points move from the source to the sink. This graph will increase in size, growing in height with increasing dimensionality and increasing in breadth with increasing number of possible variable settings.

When ants initially move through the graph during the first phase, they randomly make decisions at each node to select the next vertex based on pheromone deposits on a given edge. Actual ants traveling deposit actual pheromones on the ground which will, over time,

evaporate. This pheromone serves to encourage ants to take this path in the future. As a path is traveled more frequently, more pheromone will be deposited, further encouraging future ants. Over time, clear paths will form because the ants are following a strong pheromone path. This behavior is mimicked in Dorigo's ACO using virtual pheromones to encourage virtual ants to choose a given edge. The virtual pheromone is stored in a matrix. The higher the pheromone concentration, the more likely an ant is to choose that edge. For certain ACO implementations, ant fitness will be evaluated at each node, though this is not necessarily a requirement. regardless of the ACO implementation, an ant's fitness is used to determine how much pheromone will be deposited along its path. The version implemented for frontier finding will construct a scenario by choosing all variables and then evaluating the function. This function evaluation ends phase one.

The second phase of ACO involves updating pheromone deposits. The pheromones are stored in a pheromone matrix which is identical to the adjacency matrix used to identify which nodes are connected to each other, but the values in a pheromone matrix instead correspond to the pheromone deposited on that edge. At the beginning of pheromone updating, some pheromone evaporation occurs. Here, the pheromone matrix is multiplied by a constant, which is intended to represent actual evaporation of pheromones. After evaporation, ants will move through the graph depositing pheromones along the path they took that is proportional to their relative fitness value. The relationship in Equation 24 is generally used when the objective function is being minimized. If one were instead maximizing an objective, the numerator in Equation 24 could simply be flipped.

$$\text{Pheromone Change}_i = \frac{Y_{avg} - Y_i}{Y_{avg} - Y_{worst}} \tag{24}$$

While this method can be used to update the pheromone, some possibility of an ant choosing a given path should be maintained. For this reason, after pheromone updating takes place, a minimum pheromone threshold must be enforced, allowing one to distinguish between a very unlikely choice and an invalid choice. This pheromone update is important because it helps guide ants to choose a given edge when moving through the graph. The more pheromone on a given edge, the more likely it is an ant will choose it.

The selection of an evaporation coefficient is difficult and often problem specific. If one were to choose an extremely high value, the method would tend to approach a hill-climbing or gradient based optimization. This would occur because very little information would pass from a generation to the next, and unless good ants chose a specific path on the previous generation, it would likely approach the minimum pheromone level. If, however, one chose a very low evaporation coefficient, unfavorable solutions would require a much longer time to be removed and convergence would be slowed significantly. Therefore, a delicate balance must be obtained and this balance is specific to the problem at hand. If addressing high-dimensional, severely multi-modal systems, one would likely want to choose a low pheromone evaporation coefficient. If, however, one were solving a relatively simple system or a nearly unimodal one, a much larger value could be chosen and the system would converge more quickly.

The third phase of ACO is to implement any necessary daemon actions. In this phase, various possible actions can be taken. In some situations users may wish to encourage diversity. For this reason, the highest pheromone paths can be removed or can undergo more evaporation. Similarly, users may wish to instead encourage movement in other paths by increasing their pheromone levels. This phase is very problem specific and may be neglected completely if desired.

ACO was initially used to solve the TSP.[61] It was refined and has been used by a variety of groups to solve problems ranging from the vehicle routing problem[78] to layout problems[204] to schedule problems[151] to a variety of other NP-problems. Additionally, ACO possesses the ability to evaluate both static and dynamic graphs. While not particularly useful for frontier finding, this means weights and vertices can be changed or removed during the iteration to represent physical changes. This is especially useful in networking problems and has been used in general problems to improve convergence properties.[134] ACO has been used to solve multicriteria problems, including aerospace concept and technology spaces [214], telecommunication optimization problems[62], and portfolio planning[58], among others. ACO has been shown to solve problems such as TSP more efficiently than other meta-heuristic methods including GAs and Simulated Annealing (SA).[62, 151]

5.3.1 Ant Colony Optimization - Multi-Objective

Ant Colony Optimization - Multi-Objective (ACO-MO) is an adaptation to Dorigo's ACO, much like NSGA-II is an adaptation to GAs. Because of the relative "newness" of these methods, many of them actually reference Deb's NSGA-II and use the nondominated sorting and crowding distance calculations to determine rank. Some more commonly used implementations include Angus' population-based ACO, referred to as a Multi-Objective Function Optimization (MOFO)[11]. Other common implementations include Guntsch & Middendorf [86] and Doerner et al. [59], though each of these are essentially a variant of an elitist ACO which uses nondominated sorting to determine which candidates should update the pheromone matrix. After consulting each of these, the author adopted an approach where some percentage (generally between 10% and 50%) of the elitist population would be used to update the pheromone matrix. This was shown to have good convergence properties and will be used throughout the study.

5.3.2 Continuous Ant Colony Optimization

Continuous Ant Colony Optimization (CACO) is a continuous adaptation of ACO and has been proposed by many authors. This approach serves to convert the pheromone matrix, described earlier, into a continuous distribution. Several approaches have been proposed, including one by Pourtakdoust and Nobahari [176], where the authors propose converting the pheromone matrix to a normal distribution whose standard deviation is initially high and which is updated over time based on the ants' performance. While an interesting concept, this is a limiting case for Dorigo's initial algorithm which allows for the possibility of having multiple paths to good solutions. If this were the case and Pourakdoust and Nobahari's extension were implemented, the method would not really converge on a given point and would instead have some large standard deviation trying to essentially average these two promising candidates. Other approaches, mentioned by Blum in his survey of ACO methods[24], have also been created, but again lack the "essence" Dorigo was striving for with the pheromone matrix. One approach, proposed by Socha [198], takes an approach much more similar to Dorigo's original intent. In this paper, Socha uses the various ants'

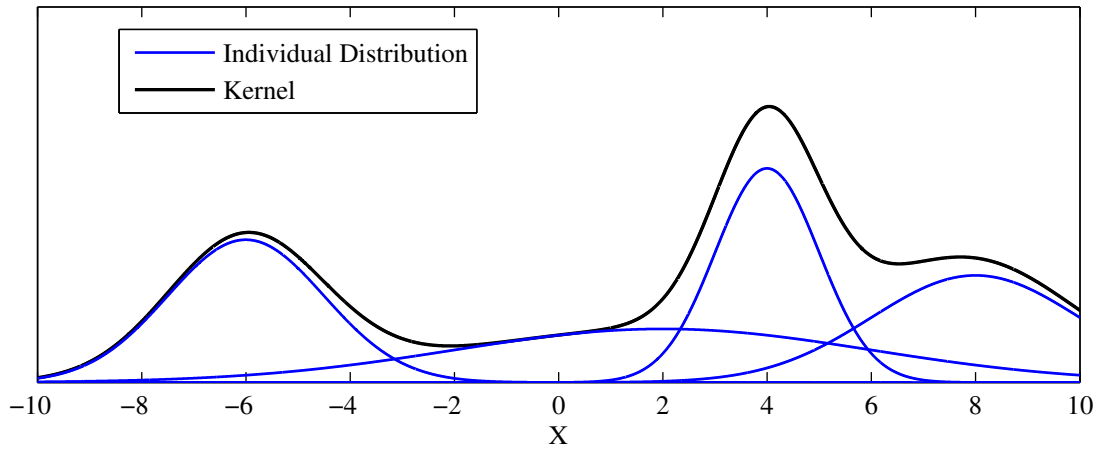


Figure 58: Illustration of Multiple Normal Distributions Forming a Kernel.

pheromone values to create a normal distribution, but instead of having a single distribution, there are a variety of distributions. These distributions form a kernel. While sampling from the kernel may not be a trivial task, Socha proposes some approximation. An ant will instead choose one of the normal distributions to follow. This allows for the possibility of multiple promising solutions, but again is somewhat limiting. This approach is illustrated in Figure 58.

In order to address this simplification proposed by Socha, the author proposes to instead interpolate among grid points. The 1-dimensional kernel shown in Figure 58 can be integrated to form a Cumulative Distribution Function (CDF). This CDF can then be evaluated based on a given probability. This scheme was adopted rather than choosing a single distribution to move along. An ant can have any value, but only a grid of points are maintained for the kernel. The pheromone from a given ant is proportionally deposited on the nearest nodes based in the inverse of its distance from a node (meaning if the ant is on a node, all of its pheromone goes to that node, and if it is halfway in between, half goes to each node, etc). This is illustrated for a 2-D grid with 5 settings per variables and 3 ants in Figure 59.

The computational expense in this method lies in the discretization of the kernel. If the kernel has an extremely large number of samples, the grid can more precisely guide ants to converge on good solutions. While this is beneficial because comparatively few

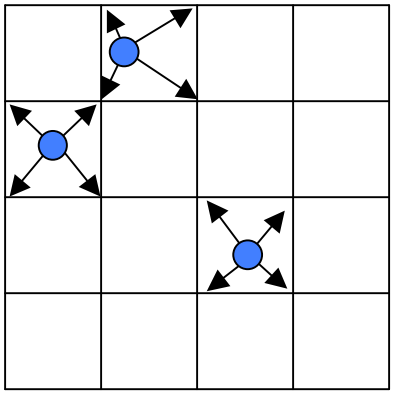


Figure 59: CACO Pheromone Kernel Updating Illustration for a 2-D Grid.

generations would be required, an excessive amount of computer memory is required to maintain, reference, and update the matrix, making each generation take an extremely long time. Conversely, if one were to use an extremely small grid with only a few settings, the grid could be used very efficiently because of its relatively small size, but it would less precisely guide the ants to favorable solutions. The selection of an appropriate discretization is problem dependent and will likely have a significant impact on convergence for certain classes of problems.

5.3.3 Ant Colony Optimization Implementation for Frontier Finding

The pheromone kernel interpolation scheme discussed in the previous section was combined with the multi-objective formulation discussed earlier for ACO-MO. This scheme was implemented in a similar fashion to that used for the Frontier Finding Real-Coded Genetic Algorithm (FFRCGA). A "segregationist" approach was adopted because, as was the case with the FFRCGA, pheromone paths were not updating as well as they could have been. Separating the population into a series of sub-populations of ants have having each sub-population of ants working with their on pheromone kernel produced faster convergence than having all ants working from a single pheromone kernel. This Frontier Finding Continuous Ant Colony Optimization (FFCACO) was successfully implemented in MATLAB and its performance will be assessed relative to the other metaheuristic frontier finding methods.

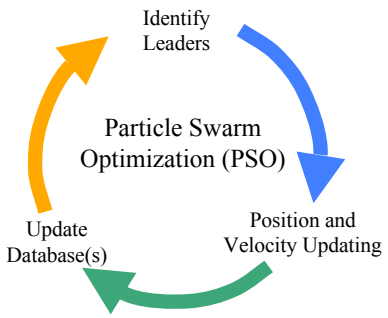


Figure 60: PSO Iteration Cycle.

5.4 Particle Swarm Optimization

PSO is a metaheuristic developed by Kennedy and Eberhart at Purdue University in 1995 which was developed with the intent to mimic bird flight.[118] During its development, the founders realized that the point masses moving across the input space came to more closely resemble insects in flight, and the algorithm came to be known at PSO. This method was developed with simple cellular automata. It improves solutions by taking an Artificial Intelligence (AI) approach, stating that individuals (or in this case cellular automata) gain their intelligence not only through their own actions but also from those of their community.

The algorithm decomposes the population into a series of neighborhoods (groups), where each member is a point mass with a position and velocity. The algorithm follows the cyclic pattern illustrated in Figure 60. In this cycle, the first step involves identifying the leader in each neighborhood. Once the leader has been identified, each member of the neighborhood updates their velocity based on their current momentum, their personal best location (x_{pBest}), and the location of their leader (x_l). This is illustrated in Equation 25.

$$v^{i+1} = v^i\omega + \phi_1(x_{pBest} - x^i) + \phi_2(x_l - x^i) \tag{25}$$

Where:

- v^{i+1} is the updated velocity
- v^i is the previous velocity
- ω is the mass coefficient for each particle

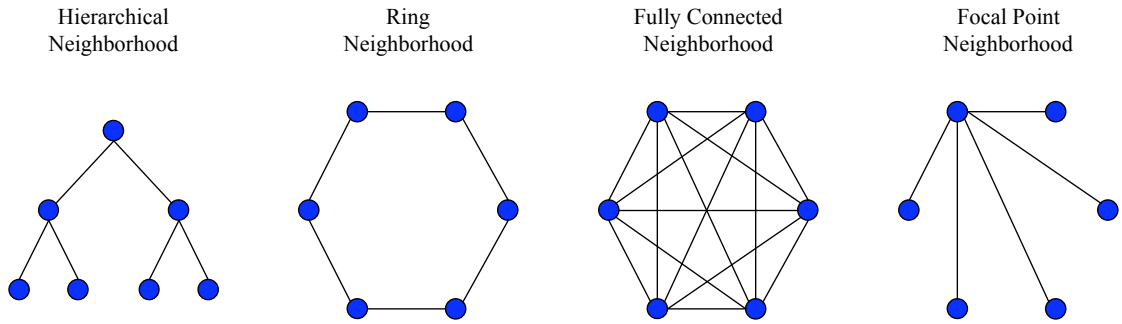


Figure 61: Example PSO Neighborhood Topologies.

- ϕ_1 is the weighting coefficient assigned to the personal best location
- x^i is the current particle location
- ϕ_2 is the weighting coefficient assigned to the leader's position

This updated velocity is then used to calculate the particle's new location at the end of this iteration. The momentum term ($v^i\omega$) is used to help "damp" the point while moving around the space. If $\omega = 0$, this term falls out and the particle may simply oscillate around the optimum solution. Similarly, ϕ_1 and ϕ_2 are used to help promote diversity in the space and to guide the particle towards the best solution it can see, respectively.

The selection of a hierarchy for the population neighborhoods is something which has been examined by many. Topologies ranging from hierarchical schemes to ring schemes to completely connected schemes or focal point schemes have been considered. The selection is often problem dependent and has a strong impact on convergence. Some example topologies are provided in Figure 61 For frontier finding, a fully connected neighborhood was chosen; the specifics for its implementation will be discussed in more detail later.

The third phase for PSO is a variant proposed in many texts which involves maintaining external databases of best solutions. Because all particles in PSO are moving, their performance is constantly changing. Therefore, a solution which is on the optimum will actually move off of the optimum because of the velocity it had when approaching it. For this reason, an external database may be beneficial. Position and/or velocity information can be stored in an external database and used when determining leaders for a given neighborhood.

This has been shown to improve convergence and will be implemented in a frontier finding variant of PSO.

Unlike the other methods discussed, PSO was initially intended to function as a continuous, or real, parameter model. Adaptations have been proposed to convert it to a binary coded algorithm, but these are not nearly as commonly used as the continuous version. Continuous PSO has been implemented for a wide variety of problems ranging from optimization to weight calculation for neural networks.[118]

5.4.1 Multi-Objective Particle Swarm Optimization

Multi-Objective Particle Swarm Optimization (MOPSO) has been developed by many using a wide variety of techniques. This algorithm differs fundamentally from methods such as GAs and ACOs because an elitist strategy cannot be implemented. Additionally, a given solution continues to move (change in position or input vector), whereas population members in GAs and ACOs would have a fixed input vector. For this reason, a variety of different scenarios have been proposed with varying success to solve multi-objective optimization problems with PSO. These schemes include aggregating functions, lexicographic ordering, sub-population approaches, Pareto-based approaches, and hybrid methods. [180]

The first of these, aggregating functions, is a concept much like an Overall Evaluation Criterion (OEC) used in first-order systems engineering applications. This approach has been implemented for solving MOPSO problems by Parsopoulos & Vrahatis [173], among others. These approaches use a form, such as the one provided in Equation 26, to aggregate a variety of different inputs.

$$Y_{OEC}^{i} = \sum_{k=1}^{n_a} \frac{Y_k^i}{Y_k^{BL}} + \sum_{k=1}^{n_b} \frac{Y_k^{BL}}{Y_k^i} \tag{26}$$

Where:

- Y_k^i represents the k^{th} output for sample i
- Y_k^{BL} represents the k^{th} output for some baseline sample sample
- n_a is the number of outputs which should be maximized

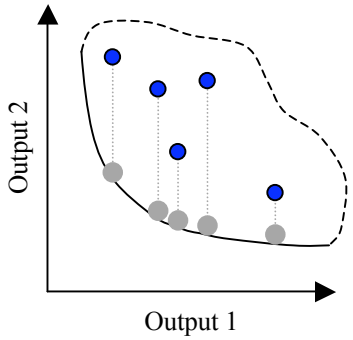


Figure 62: PSO Lexicongraphic Ordering Illustration.

- n_b is the number of outputs which should be minimized

While an approach such as aggregation does allow one to address the trade-off between one output and another, in effect penalizing a solution if it performs poorly on a given output, this method would not be helpful in identifying a boundary in a given dimension. Instead it would provide a few points which may be on the boundary. For this reason, aggregate approaches were not considered any further.

The second approach, lexicongraphic ordering, is a method that can be used to identify a frontier by "dropping" points onto the frontier. This is illustrated graphically in Figure 62. In this scheme, a population will be initialized and its neighborhood will be identified based on proximity to a *single* output variable (in Figure 62 this is Output 1). The points are then only allowed to move in a single dimension towards the direction for improvement (down in Figure 62). Once these points have reached the boundary (shown with grey points), the algorithm will terminate. If more than 2 output variables were being maximized or minimized, then the points would "fall" towards a 3^{rd} frontier. This continues until the points have "fallen" against all frontiers except for the one initially used to create neighborhoods. This scheme has been advocated by some in the literature, including Hu & Eberhart.[110] While this approach does tend to work well for solving low-dimensional MOPSO problems, it has been shown to degrade in performance as the number of outputs used increases.[180] Because this method would be necessary to solve moderate- to high-dimensional problems, it will not be further considered.

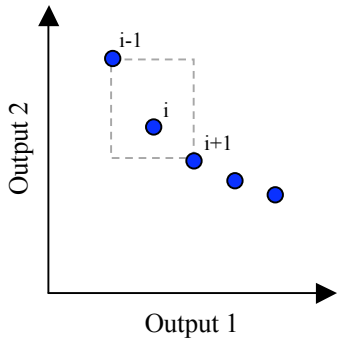


Figure 63: PSO Equivalent Distance Calculation.

Sub-population based approaches divide the population into several sub-populations and then has each of these sub-populations maximize or minimize a given output. After a period of time, these various sub-populations then recombine and exchange this information in an effort to promote diversity and find better solutions to the multi-objective problem. This method has been implemented by several, including Parsopoulos et al.[172], to solve multi-objective problems. This approach is much like the "segregationist" approach described for the FFRCGA and FFCACO earlier in this chapter, with the exception of information exchange; however, it would likely produce "clumps" of points at the corners of the space rather than having a more evenly distributed population along the frontier, as would be the case with a Pareto-based approach. For this reason, it will not be examined further.

Pareto-based approaches all function by identifying non-dominated solutions and promoting diversity along this front of non-dominated solutions. Many of these approaches, like that implemented in Deb's NSGA-II, function by first doing a non-dominated sorting of the population and then promoting diversity as a secondary objective. The most effective schemes commonly use an "equivalent area" calculation to find how much relative area on the Pareto front a given solution occupies. This is graphically illustrated in Figure 63, and is equivalent to the crowding distance calculation proposed by Deb in his 2002 paper[50], which was provided in Listing 5.1 earlier.

Other methods include the assumption that a given point is dominant over all points within some hypersphere or hypercube surrounding the point. This prevents points from clustering in a given area and promotes diversity on the front of nondominated solutions.

Listing 5.2: Pareto-Based MOPSO Using an External Database.

```
function[] = mopso()
initialize population position, X
initialize population velocity, V
initialize position external database, Ydatabase

while convergence not reached
    calculate population fitness, Y
    read in Ydatabase
    perform nondominated sorting of [Y Ydatabase] to ID 1st front
    leaders = nondominated population members
    if | leaders | allowable size
        perform equivalent area calculation
        leaders = leaders with highest rank
    end

    update V, Y
    write leaders to Ydatabase
end
```

These approaches are much like that proposed in the first Nondominated Sorting Genetic Algorithm (NSGA) in that the size of the hypersphere or hypercube is extremely problem dependent and will likely have a significant impact on convergence.

In ACO and GA, promising information about the front is maintained within the population either directly or through the use of a pheromone matrix. Because candidate solutions for PSO are moving, the front is also constantly moving. One adaptation to mitigate these potentially negative effects is to maintain an external database. This database is a catalogue of the leaders for a given population, and in the case of a MOPSO implementation, this would consist of points on the nondominated front. It has been shown in the literature that the size of this database should be capped, meaning it can only become so large, to improve convergence properties.[180] In such a scheme, the properties of an elistist GA or elitist ACO could be mimicked, as illustrated in Listing 5.2. The procedure was created by the author but is based heavily on the Kennedy & Eberhart text.[118]

After initialization of the position, velocity, and external databases, the algorithm is surprisingly straightforward. After evaluating the population's fitness, this fitness is combined with the external database. Then, a single nondominated sorting is performed of this aggregate. All nondominated solutions, identified in the nondominated sorting, are

leaders, unless there are too many leaders. Then, an equivalent area calculation, much like Deb's crowding calculation provided in Listing 5.1, is performed to choose the most diverse nondominated solutions, who will then become the leaders. This is the method's benefit over a Pareto-based GA or ACO implementation. In either a GA or ACO implementation, all solutions' ranks are calculated (or at least a sufficient number to identify the best half of the population) and then a crowding distance calculation is performed if necessary. This requires repeatedly performing a nondominated sorting (which is the most expensive portion of an NSGA-II generation) several times. This process is also the most expensive in a Continuous Ant Colony Optimization - Multi-Objective (CACO-MO) implementation, but the cost of pheromone matrix operations is a close second, depending on the number of settings in the pheromone matrix. Instead, a Pareto-based MOPSO only requires a single nondominated sorting, making it more efficient on a per-generation/iteration basis.

Additional adaptations have been proposed to mimic the GA mutation operator by including a "turbulence" factor applied to the velocity calculation. This is a random variation intended to cause points to fly in a slightly different direction which will promote diversity and may improve convergence.

Pareto-based MOPSO schemes similar to the one outlined in Listing 5.2 have been proposed by many, including Bartz et al.[14] and Reyes & Coello [179]. These approaches commonly involve using some sort of equivalent distance or crowding calculation as well as external databases and possibly GA-like operators to improve convergence.

The final commonly seen type of MOPSO seen in the literature is a hybrid method. These methods commonly combine aspects of Pareto-based dominance with aggregate functions or other methods in an effort to avoid local optimum solutions and improve global search capability. One such approach, published by Mahfouf et al.[141], proposes expanding PSO to have an acceleration term and also uses GA-like operators in an effort to improve convergence.

Based on this survey, most authors tend to use a Pareto-like dominance scheme. There is a great variety of literature on the subject available. Interestingly, this scheme most closely resembles that used by FFRCGA and FFCACO. For this reason, a Pareto-like scheme,

similar to that proposed in Listing 5.2, was adopted.

5.4.2 Particle Swarm Optimization Discrete Optimization

PSO has been used for continuous optimization in a wide variety of studies. This approach is most commonly used; however, the method was initially formulated for a binary decision problem.[118] This implementation was much like the continuous version, as described in Figure 60, but with one exception. After calculating the new position, a rounding was done where all positions were rounded to either 0 or 1. This idea was extended by the author to use for solving discrete problems. The difference is that instead of rounding the final positions to either 0 or 1, they will be rounded to one of the possible discrete choices which can be made. This modification will allow a user to use PSO to solve discrete problems, but one should be cautious when limiting a particle's ability to move throughout the space. If the move limit is too great relative to the distance between choices, then the particles will be unable to move to another choice even though their velocity vector is pointed in that direction. Through experimentation, the author has found that setting the move limit to be 225% of the maximum distance between discrete choices allows enough freedom to let particles move from one bin to another, but at the same time limits movement enough to prohibit particles from oscillating between one end of the input space and the other.

5.4.3 Frontier Finding Particle Swarm Optimization

In order to modify the Pareto-based MOPSO identified in the previous section for frontier finding, a procedure like that adopted for FFRCGA and FFCACO was adopted. Several "segregated" sub-populations were created within the algorithm. Each of these sub-populations then seeks a different Pareto frontier. This method was implemented in MATLAB, much like the process developed for FFRCGA and FFCACO, producing a Frontier Finding Particle Swarm Optimization (FFPSO) algorithm.

5.5 Comparison to Additional Methods

Up through this point, three different algorithms have been discussed: a GA, an ACO, and a PSO. There are a multitude of other methods which can be considered, but the purpose

of this section is to show that while these methods will not be implemented, nearly all are similar to one of these three. Therefore, while these additional methods may not have been implemented, some measure of their performance can be deduced from at least one of these three.

The Artificial Bee Colony (ABC) is a metaheuristic method developed by Nakrani and Tovey in 2004.[160] The algorithm is intended to mimic bees foraging for pollen. Various bees explore the problem space and, after returning to the hive, do a "waggle dance" to encourage other bees to follow them back to the pollen source. The strength of the "waggling" is proportional to the fitness of the pollen source. ABC was initially applied to dynamic server allocation but has also been used to solve a variety of other problems showing performance that is competitive with algorithms including RCGA and PSO.[115, 227] A very similar algorithm, Virtual Bee Algorithm (VBA), was proposed later that year by Yang; this was initially used to solve both continuous and discrete engineering optimization problems. [226] Bee algorithms such as ABC and VBA are very much similar to ACO. The "waggle dance" in these algorithms is remarkably similar to the pheromone matrix used in ACO. For this reason, one could say that in evaluating ACO's performance, one could also be conducting a first-order assessment of how well ABC and VBA would perform. Additionally, implementing ABC or VBA would require many of the same issues and modifications for frontier finding that were used for ACO. For these reasons, these methods will not be further considered.

The Firefly Algorithm (FA) is a metaheuristic algorithm developed by Yang in 2007.[227] The algorithm is intended to mimic fireflies flying in the evening sky. Fireflies have a tendency to "cluster" because they are attracted by the luminescence of other fireflies. The algorithm has two mechanisms to allow the fireflies to coalesce:

1. A firefly is attracted to brighter fireflies, but their movement toward brighter fireflies is done randomly [228]

2. The firefly's attraction to brighter fireflies is proportional to the observed brightness of that firefly (which decreases exponentially with the firefly's distance from the brighter

fireflies) [228]

In examining Yang's convergence mechanism, one can easily see that this is *very* similar to the mechanism developed by Kennedy & Eberhart in PSO.[118] One could imagine that taking into account the distance between two fireflies when determining how the "leader" (in PSO terms) would be assigned leads to the firefly population forming a variety of clusters even if fully connected, whereas a PSO implementation would always converge to a single point. This would allow one to see other candidate "good" solutions. Because of FA's similarity to PSO, one could say that evaluating PSO's performance would also serve as a first-order assessment of FA performance. If one were to implement FA for frontier finding, one would likely make many of the same modifications that were made to PSO. For these reasons, FA will not be considered any further.

Another promising method which has not been previously discussed is SA. This algorithm was published by Kirkpatrick et. al. in 1983 and draws from metallurgy and crystal formation to find the global optimum.[121] Another useful analogy for describing SA would be dropping a series of "balls" on some space. The height the balls bounce is dependent on the energy in the ball (kinetic and potential). This energy dissipates over time, so the ball will not bounce as high as it previously had. After some point, the balls will stop bouncing and "roll" into the local optimum nearest them. This is very much like SA, but the energy of the various "balls" is instead driven by the virtual "temperature" in the SA implementation.[227] While this method has been proven to converge upon the global optimum, its convergence rate on this solution is highly dependent on the virtual "temperature" schedule.[227] Additionally, it has been shown by some to have a poorer convergence relative to algorithms such as GA and ACO.[214] If one were to consider SA for frontier finding, it would likely require modifications similar to PSO to store favorable solutions and enforce penalty functions. For these reasons, SA will not be considered.

5.6 Conclusions

In this chapter, metaheuristic optimization methods were considered for finding the frontiers of a space. Three favorable methods were identified, specifically GAs, ACO, and PSO.

The modifications for these algorithms that were necessary to find the frontiers of a space were discussed. These include modifying the algorithms (if necessary) to solve continuous problems, modifying them to solve multi-objective problems, and segregating the population into a series of subpopulations, each of which would search for a given nondominated front. Each of these modifications have been documented in the literature individually (for example, an RCGA, a MOPSO, or a CACO), but this author was unable to identify any sources in the literature which implemented all of these, much less any sources that implemented all of them to search for the boundary of the space. The following chapter will discuss implementation specifics on how these algorithms will be used to find the frontiers of a space and therefore allow the user to quantify the epistemic model uncertainty present in a given analysis or series of analyses.

CHAPTER VI

METAHEURISTIC FRONTIER FINDING ALGORITHM IMPLEMENTATION FOR CAPTURING EPISTEMIC MODEL UNCERTAINTY

6.1 Introduction

In this chapter, the three metaheuristic optimization methods were selected from a literature review of metaheuristic frontier finding algorithms in Chapter 5. These algorithms will then be implemented in experiment 3 to determine which can most quickly approximate the bounds of the space found with the full factorial and also determine the best algorithm's performance relative to the space-filling Design of Experiments (DoE) performance discussed earlier in Chapter 4.

Before assessing their performance, issues associated with restarting the algorithm will be addressed. Specifically, these algorithms are used to solve Nondeterministic Polynomial (NP)-hard problems quickly, but their methods for doing so prevent one from having complete confidence that the result is the best solution. In order to address this, experiments will be conducted to identify how many times the algorithms should be restarted and how long each should be allowed to run before being restarted. After this restart question has been addressed, each method will be implemented for the three test problems used earlier in Chapter 4. Based on these findings, the best method will be further examined and a restart rule will be developed, allowing users to in essence know how long they should let the algorithm run before restarting as a function of the number of input variables.

6.2 Metaheuristic Frontier Finding Restarting Considerations

Metaheuristic methods are a class of algorithms which use population-based approaches with stochastic policy transitions to quickly approach promising solutions to NP-hard problems. While these approaches do generally converge quickly, the stochastic nature of their policy transitions requires that some restarting be performed in order to gain some confidence that

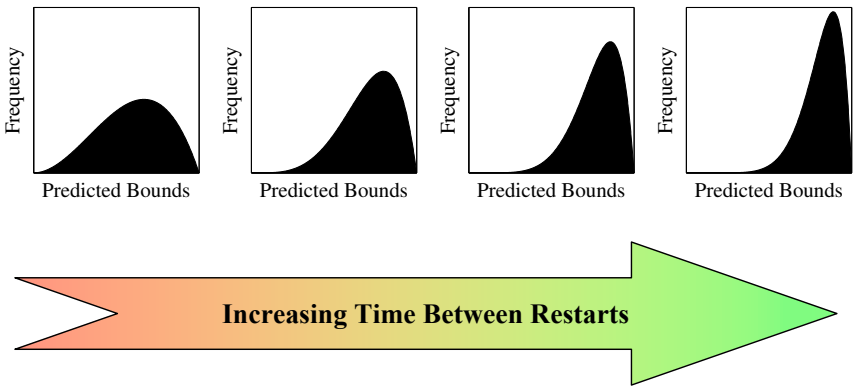


Figure 64: Illustration of Trade-off Between Long and Short Run Time When Capturing the Bounds.

the bounds being captured are the true bounds, or are at least near the true bounds.

When restarting these algorithms, there is a trade-off between the amount of time the algorithm is left to run and the number of restarts. This trade-off is illustrated in Figure 64. If one restarts the method very frequently, a large number of samples will be taken. However, these samples will have a relatively low confidence of reaching the true bounds of the space. If, however one were to let the algorithm run much longer, one would have a greater confidence of reaching the boundary of the space (distribution skews to right), but for the same amount of time there would be fewer samples.

In order to quantify this trade-off and identify a relative convergence criteria for these methods, a series of tests were conducted. Because some restarting is required to capture the bounds of the space, the following two possibilities were envisioned:

1. Samples consistently find the same extreme bounds (the bounds have been found).

2. Samples do not consistently find the same extreme bounds (the bounds may not have been found).

While the first possibility is of course the most desirable, it may not always be possible. It was assumed that if at least three samples obtained the same extreme bound, then this extreme bound would be the true bound. For this reason, the second possibility was examined. After so many samples had been taken, the sample mean, $\bar{\mu}$, and sample standard deviation, $\bar{\sigma}$, could be calculated. If a sample exists which is likely not on the bound

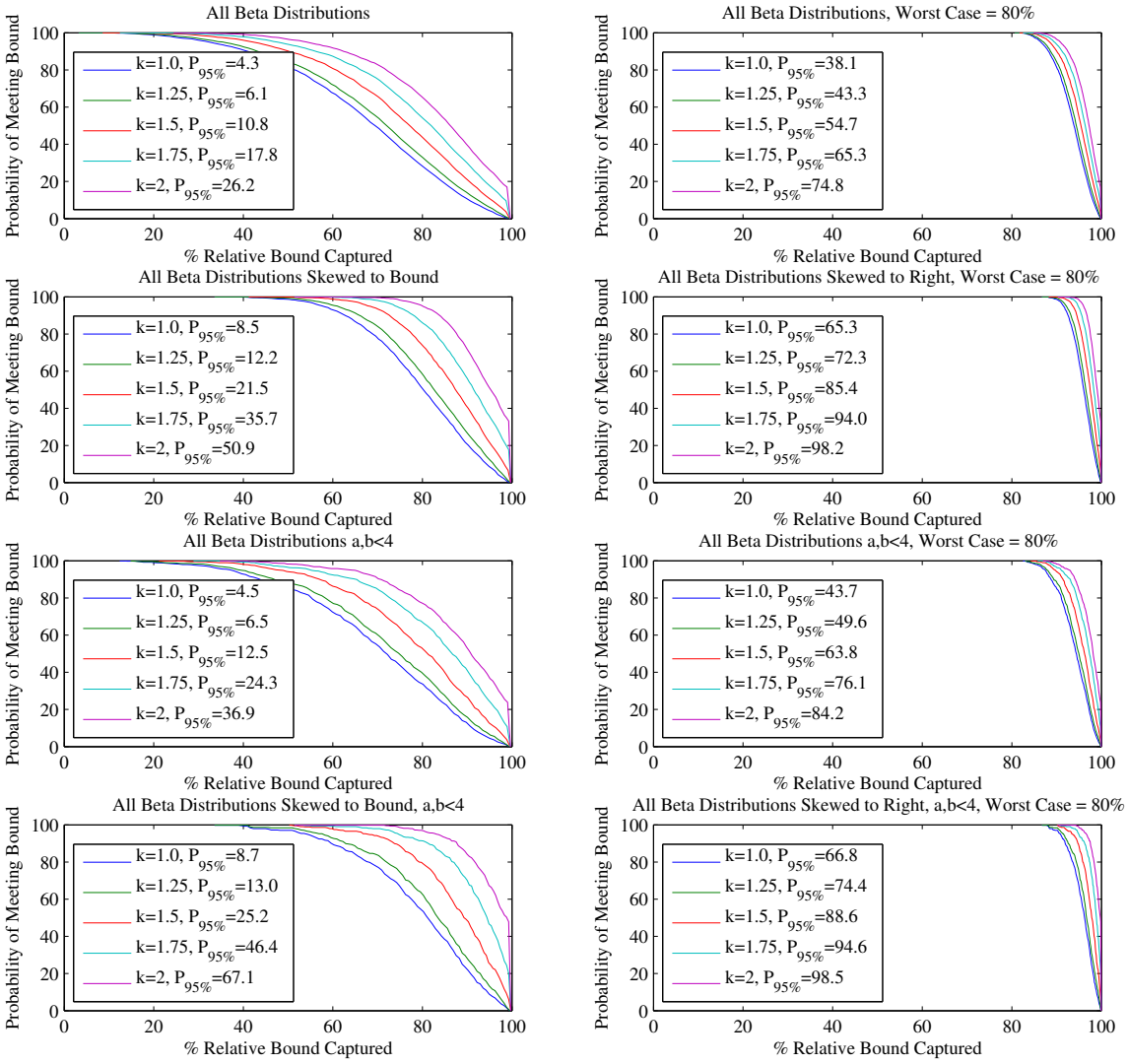


Figure 65: Resulting CDFs for Various Beta Distributions.

(meaning there are not several samples at the same extreme point) and is more than $k\sigma$ from the mean bound, then one could have some moderate confidence that this sample is near the bound and terminate. In order to identify an appropriate value for k, a series of tests was completed. A large variety of beta distributions was created in order to determine what k should be. In this experiment, a full factorial of beta distributions with $a, b \in [1, 8]$ was created and the amount of the bounds that were captured, given a variety of settings of k was determined. These results are provided as a Cumulative Distribution Function (CDF) in the top left pane of Figure 65.

When examining this figure, one can see that regardless of the k selected, there is a

very low probability of capturing 95% of the bounds. However, a variety of assumptions can be made to provide results more representative of one would see from a frontier-finding algorithm. For example, the top left pane assumes a worst-case possibility of not capturing any of the bounds, which does not physically make sense for frontier finding algorithms; one would expect that even if the method were to terminate much too quickly, it would still capture some portion of the space. After examining data from each of the methods, no method captured less than 85% of the space, regardless of when it terminated. For this reason, 80% was taken to represent a "worst-case" scenario. This is represented in the top right pane of Figure 65. Additionally, after examining the data, one realized that in the distributions of the bounds captured that were created by each of these frontier finding methods, the distribution is always both skewed to the bound (right if considering an upper bound) and is not a "peaky" distribution, meaning that the equivalent beta distribution would be $a, b \in [1, 4]$. These subsets of the full factorial data on the top row of panes in Figure 65 are provided in the second row (only skewed distributions) and third row (only non-"peaky" distributions), where the left column has a worst case possibility of capturing 0% of the space and the right column represents a worst case possibility of capturing 80% of the space. Finally, if one were to look at the only cases meeting both the skewed and non-"peaky" requirement, one arrives at the bottom row of panes in Figure 65. In the lower right, which is believed by the author to be the most representative of any scenario created by these frontier finding algorithms, one can see that for $k = 1.5$, one has an 88.6% confidence of capturing 95% of the bounds and nearly 100% confidence in capturing 90% of the bounds. It is true that if one were to increase k, the confidence would increase, but the number of function evaluations exponentially increases with increasing k. For this reason, $k = 1.5$ was deemed to be sufficient, and the relative convergence criteria provided in Listing 6.1 was implemented. While some may consider an 88.6% confidence of capturing 95% of the bounds somewhat insufficient, it was shown with the academic test functions to have better performance.

Once a relative convergence rule was developed, the appropriate restart time was considered. This was identified to be a function of both the problem being considered and the

Listing 6.1: Relative Convergence Criteria for All Frontier Finding Algorithms.

```
while not converged
    evaluate frontier finding algorithm
    if samples>2
        calculate mean (mu) of each upper and lower bound
        calculate standard deviation (std) of each upper and lower bound
        if three samples have same extreme bound
            convergence reached
        elseif sample exists where y>mu+1.5*std
            convergence reached
        end
    end
end
```

method being used to capture the bounds (Frontier Finding Real-Coded Genetic Algorithm (FFRCGA), Frontier Finding Continuous Ant Colony Optimization (FFCACO), or Frontier Finding Particle Swarm Optimization (FFPSO)). In order to compare relative performance, each of the three frontier finding methods were considered for the three test functions (Schwefel, Michalewicz, and Sphere) for 4 input variables.

In order to both test the fitness of the relative convergence criteria and create truth data, each test function was evaluated with each method 100 times. In order to determine the appropriate restart time, each method was allowed to run for up to 10 minutes while data was saved intermediately so any restart time 10 minutes or less could be considered.

6.2.1 Frontier Finding Real-Coded Genetic Algorithm Performance

The FFRCGA previously described was implemented for the Schewfel, Michalewicz, and Sphere test functions. The goal for this portion of the study was to identify the "best" restart time for the FFRCGA such that the maximum amount of the bounds could be captured in the minimum total amount of time. The results for the FFRCGA are provided for all three test functions in Figure 66.

In this figure, the horizontal axis corresponds to the total amount of time spent. The vertical axis is the percent relative bounds, which was calculated earlier. Grey lines in this plot are instances where the restart has not yet occurred. Colored lines represent the actual solutions (the system converges as soon as the line changes from grey to non-grey). The

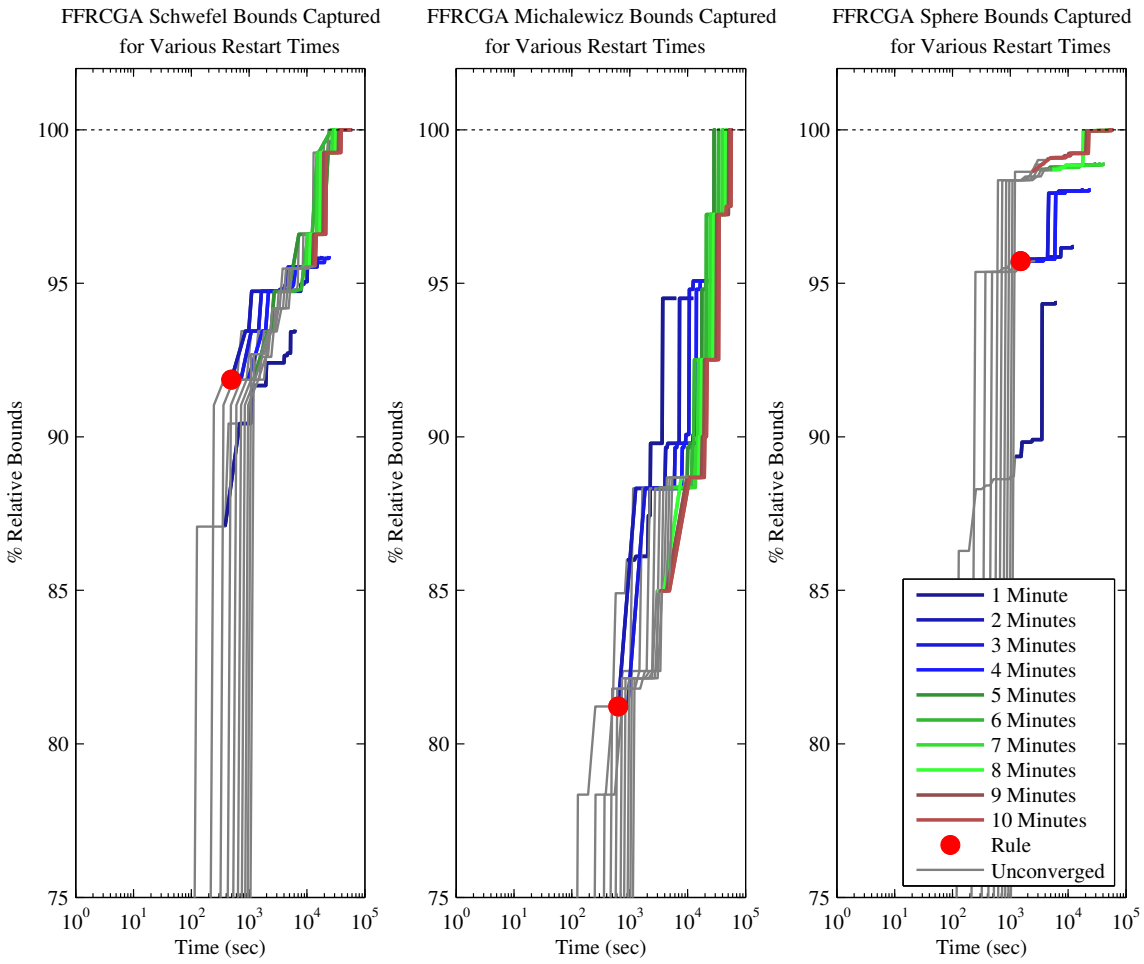


Figure 66: FFRCGA Test Function Performance.

Table 2: FFRCGA Performance with 2 Minute Restart.

Test Function	Schewfel	Michalewicz	Sphere
% Relative Bounds	91.87%	81.22%	95.72%
Total Time (sec)	489.54	630.83	1516.4
Total Time (min)	8.16	10.51	25.27
# Samples	4	5	12
Minimum σ	1.69	1.59	1.59

Schwefel, Michalewicz, and Sphere test functions are provided in the left, center, and right panes, respectively. Examining the data, it was found that the FFRCGA showed the best performance when it was restarted after 2 minutes. This point is shown in each pane with a large red dot. More detailed information for FFRCGA performance with a 2 minute restart time is provided in Table 2.

Some interesting trends can be seen from this data. First, it seems that the FFRCGA is unable to capture the bounds for the 4 input variable version of the test function. The method only captures 95% of the bounds for one test function and requires nearly 30 minutes to do so. Second, the bounds captured are proportional to their complexity (best performance for Sphere, worst for Michalewicz), but the times required are not proportional to their complexity. From this data, one can see that either the relative convergence rule should be more strict, meaning a higher value of k should be chosen, or more samples should be taken. Either of these two approaches would increase the amount of time required for convergence, which is not particularly promising.

6.2.2 Frontier Finding Continuous Ant Colony Optimization Performance

A similar experiment was conducted with the FFCACO model developed. A 4 input variable version of each test function was considered. Restart times of up to 10 minutes were considered, meaning the FFCACO would run for up to 10 minutes and then restart. The results for this experiment are provided graphically in Figure 67.

This figure was used to identify the best restart time for FFCACO, which was found to

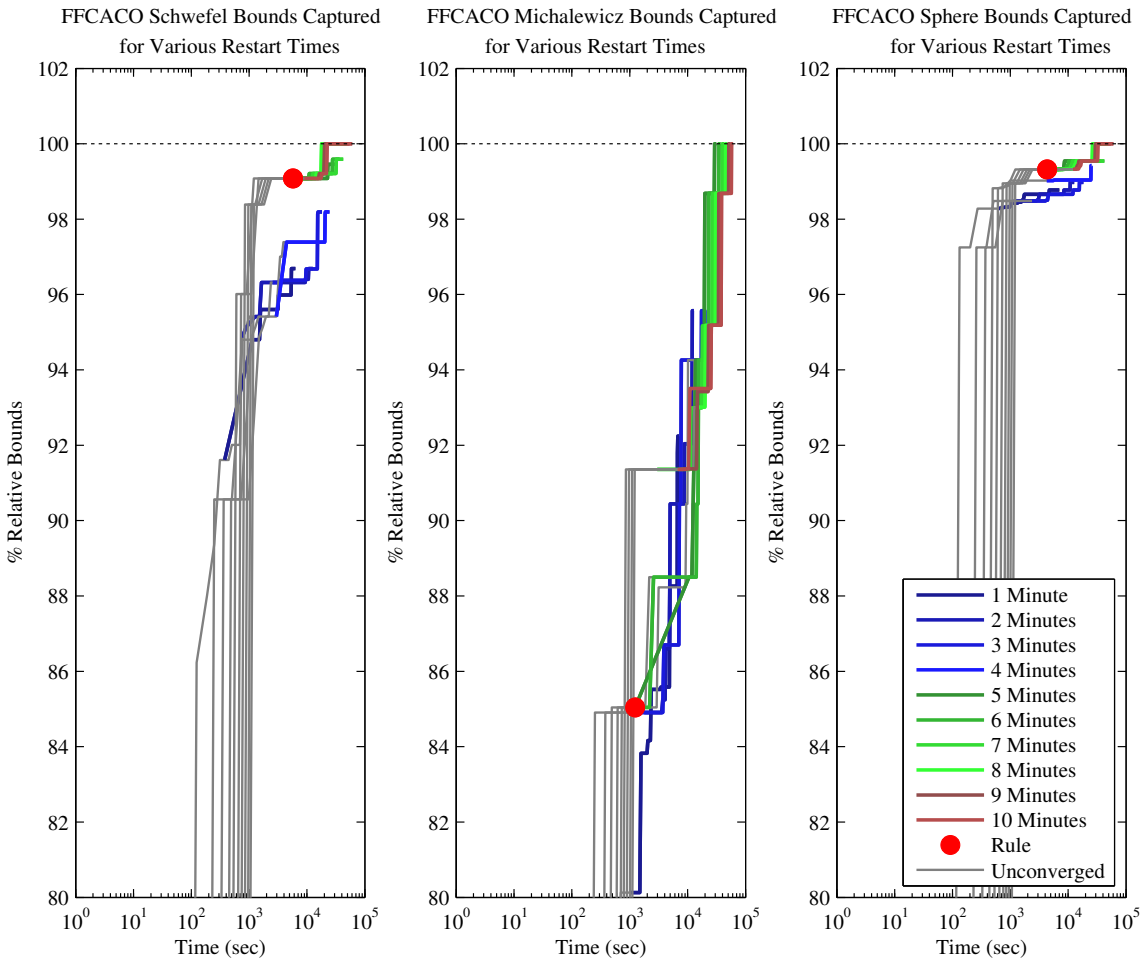


Figure 67: FFCACO Test Function Performance.

Table 3: FFCACO Performance with 5 Minute Restart.

Test Function	Schewfel	Michalewicz	Sphere
% Relative Bounds	99.08%	85.04%	99.32%
Total Time (sec)	5731.1	1241.8	4320.2
Total Time (min)	95.52	20.70	72.00
# Samples	19	4	14
Minimum σ	1.53	1.70	1.53

be 5 minutes (FFCACO should run for 5 minutes, then restart until converged). Much like the previous method, the left, center, and right panes are the Schwefel, Michalewicz, and Sphere functions, respectively. Like the previous method, the horizontal axis is the total time in seconds and the vertical axis is the percent relative bounds captured. More detailed convergence information is provided in Table 3.

In this table, one can see initially that the FFCACO does a much better job of capturing the bounds of the space than the FFRCGA, but it does so at a significant time cost. The FFCACO requires more than an hour to converge for two of the three test functions, and the Michalewicz function, which converges after four samples, does a very poor job of capturing the bounds of the space (approximately 85%). From this information, one can see that while this method does a superior job of capturing the bounds of the space, the time cost is too great. Therefore, one should consider making the rule less strict, meaning the Schewfel and Sphere functions would converge more quickly, though likely with a poorer performance capturing the bounds. While this is a possibility, it would further damage the already poor performance on the Michalewicz function.

6.2.3 Frontier Finding Particle Swarm Optimization Performance

Moving on to the FFPSO, a similar experiment was conducted. Here, the FFPSO performance was assessed when trying to capture the bounds of the three test problems for 4 input variables. Much like the previous methods, restart times of up to 10 minutes were considered and intermediate data was obtained so shorter restart times could be examined

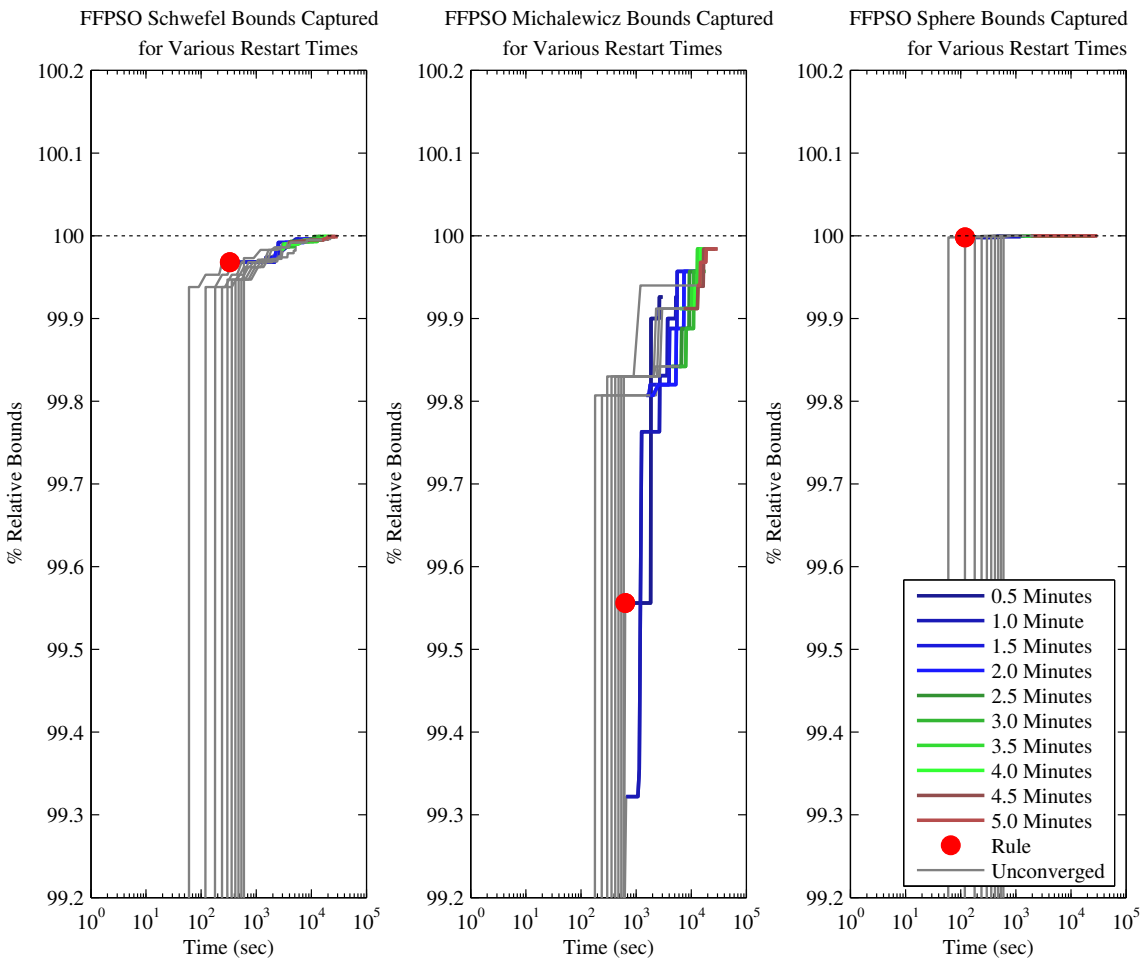


Figure 68: FFPSO Test Function Performance.

as well. The data from this study is provided in Figure 68.

Figure 68 is set up much like the previous two. The left, center, and right panes are the performance for the Schwefel, Michalewicz, and Sphere test functions. The horizontal axis in each pane is the total time and the vertical axis is percent relative bounds captured. The colored lines represent converged solutions while the grey lines represent unconverged solutions. One can see from Figure 68 that the FFPSO outperforms the FFRCGA and FFCACO, both with respect to time to convergence and bounds captured. This figure was used to establish a FFPSO restart time of 30 seconds, meaning every 30 seconds FFPSO would restart and continue doing so until the relative convergence criteria was met. More detailed information for the FFPSO is provided in Table 4.

Table 4 shows that the bounds of the space are quickly captured for all problems. Over

Table 4: FFPSO Performance with 30 Second Restart.

Test Function	Schewfel	Michalewicz	Sphere
% Relative Bounds	99.97%	99.56%	100.00%
Total Time (sec)	331.25	633.09	120.06
Total Time (min)	5.52	10.55	2.00
# Samples	11	21	4
Minimum σ	1.53	1.70	1.67

99.5% of the bounds are captured for each problem in under 11 minutes for all test functions. This shows that the FFPSO is able to perform very well with the relative convergence rule as is and that there is no further need for modification.

Some may say that the significant difference in performance between these three frontier finding algorithms is due entire to implementation and that the results could be completely reversed if implemented by others. While some benefit may be achieved, it is the opinion of the author that this difference is due more to the fundamental inner workings of the methods than their implementation. For example, consider the difference between the FFRCGA and FFCACO. The FFCACO does a better job capturing the bounds, but it not only takes more samples, but also requires more time between restarts. This is because the knowledge about good solutions is maintained by the FFCACO in the virtual pheromone matrix. This matrix is much more efficient at finding the bounds of the space, as seen in Tables 2 and 3, but it requires much more time. Manipulating and updating the pheromone matrix requires a relatively expensive interpolation. Conversely, considering the FFRCGA, the knowledge about good solutions is maintained in the vectors which represent each population member. This is a more compact way to represent good solutions, but does not work as well as the pheromone matrix.

If one were to then compare the FFRCGA and FFCACO to the FFPSO, a clear difference is seen. The FFRCGA and FFCACO are elitist population algorithms, which spend a great deal of time per generation (or iteration) performing a nondominated sorting of

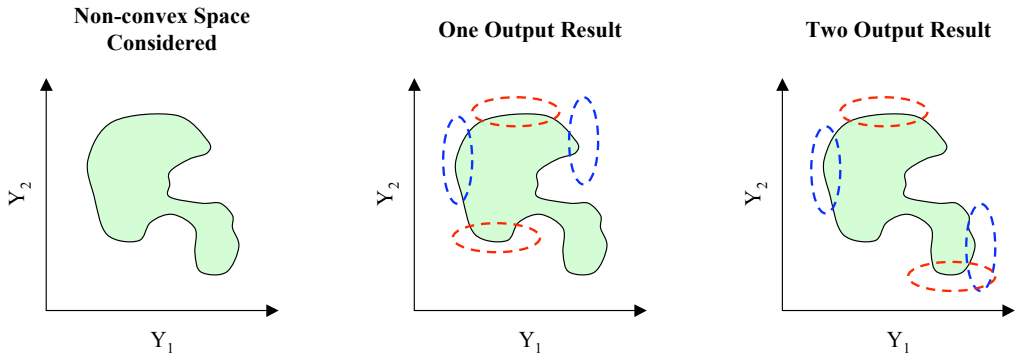


Figure 69: Illustration for Improved Frontier Finding Performance When Multiple Outputs Considered.

the population. This requires that half the members of the population be nondominated sorted. Conversely, the FFPSO does not require such information, only that the leaders must be identified. This means that only the first front is of interest, not the front number for half the population. This makes it intrinsically a cheaper operation. Additionally, the mechanisms in FFPSO can be completely vectorized, whereas operations such as crossover and mutation in the FFRCGA and pheromone matrix updating for the FFCACO must be performed for each population member, further increasing their computational cost.

Based on this information, one can say that the FFPSO clearly outperforms both the FFRCGA and FFCACO for these three test functions. Because of the clear difference in performance, only the FFPSO will be considered at this point and neither the FFRCGA or FFPSO will be considered any further.

6.3 Further Frontier Finding Particle Swarm Optimization Restarting Rule Development

In order to implement the FFPSO algorithm for a wide variety of different problems, one must identify how its convergence varies with the size of the space. Two different parameters were considered, specifically the number of outputs and the number of inputs. The number of outputs was found to have an interesting effect on convergence. Specifically, as the number of outputs increases, the relative bounds captured also increase. While this may seem initially counter-intuitive, for small and moderate dimensional non-convex problems it was found to be the case. Consider the 2-D non-convex shape provided in Figure 69.

The left pane of this figure provides the space being examined. The center pane uses red and blue dotted ovals to show the bounds which would likely be found if either Y_1 or Y_2 were examined individually. Here, we can clearly see that the entire bounds of the space aren't found because of the non-convex property of the space. The frontier finding algorithm would have the distinct possibility of not finding the "niche" in the bottom right corner because of its relatively small "opening" to the rest of the feasible space. If, however, one were to examine both Y_1 and Y_2 simultaneously, the algorithms' ability to encourage diversity would indeed initially find these bounds, but points would in effect spread out along the boundary of the space and "fall" into this small "niche" in the bottom right corner. As is shown with this illustration, examining more outputs allows the algorithm to better solve non-convex problems. If, however, one were to consider convex problems, there would not be any benefit in considering multiple outputs, but there would also not be any real harm (if anything, a time savings would be seen with elitist algorithms because there would be fewer fronts to calculate). Therefore, it is recommended that all outputs be considered simultaneously.

In order to develop a restart rule for the FFPSO algorithm developed, only single output cases will be considered. Based on the discussion provided earlier, the algorithm's performance would be poorest if only one output were considered, and the resulting rule could be thought of as a "worst case" rule. In order to assess what this "worst case" rule should be, the Schwefel and Michalewicz functions were evaluated. The Sphere function was not considered because FFPSO performance for the 4 input variable case was great enough; the two more difficult functions were thought to be a better potential "worst case" to develop a relative convergence rule. The data for 6, 8, 10, and 12 input variables are provided for the Schewfel and Michalewicz functions in Figures 70 and 71, respectively.

From these two figures, relative convergence rules were adopted as a function of the number of inputs. The relative convergence rule is provided in Figure 72.

Using this rule, the red dot representing where FFPSO would converge is provided in Figures 70 and 71.

Tables 5 and 6 show more detailed information for FFPSO for 6 to 12 input variables

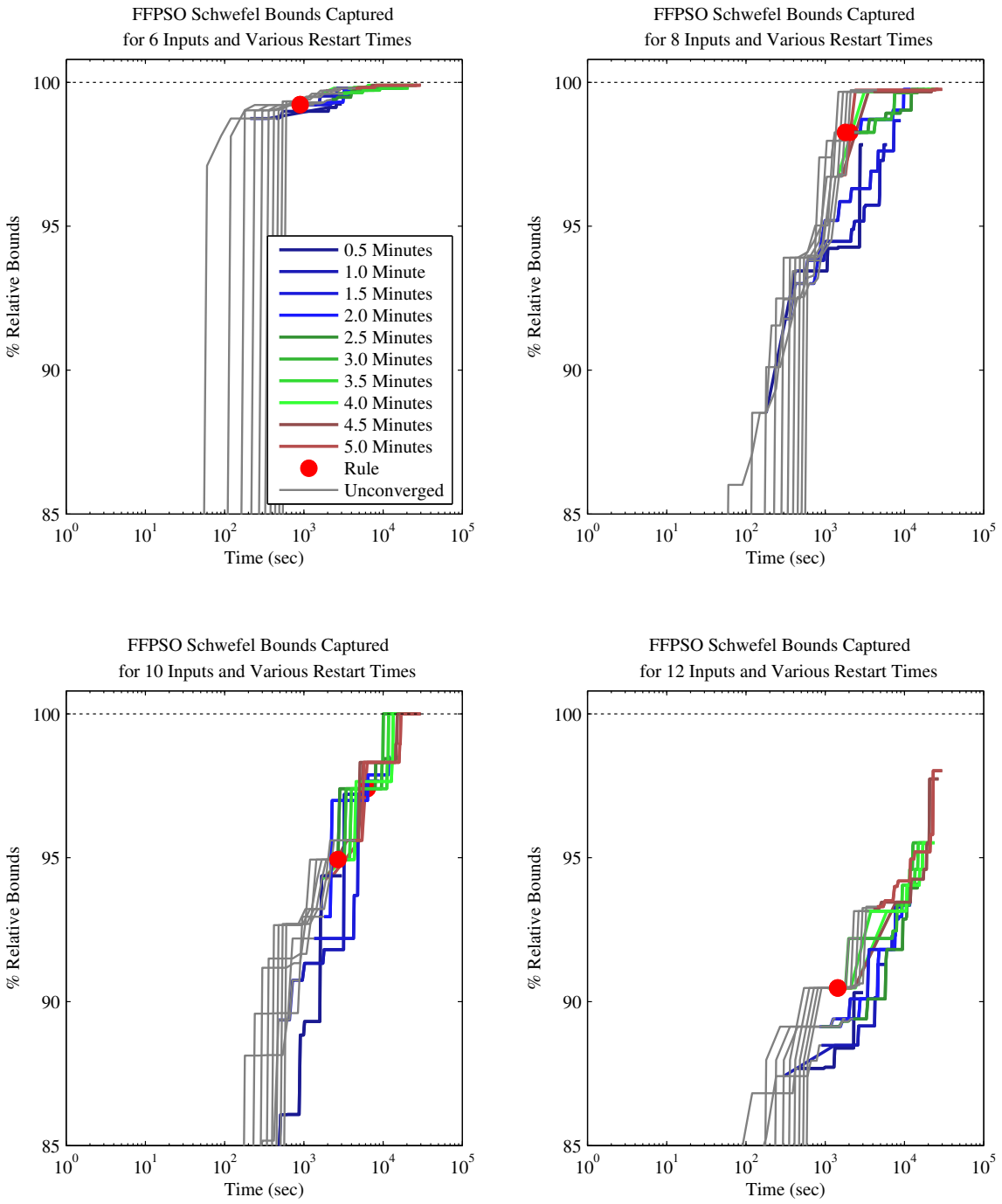


Figure 70: FFPSO Test Function Performance for Schwefel Function for up to 12 Inputs.

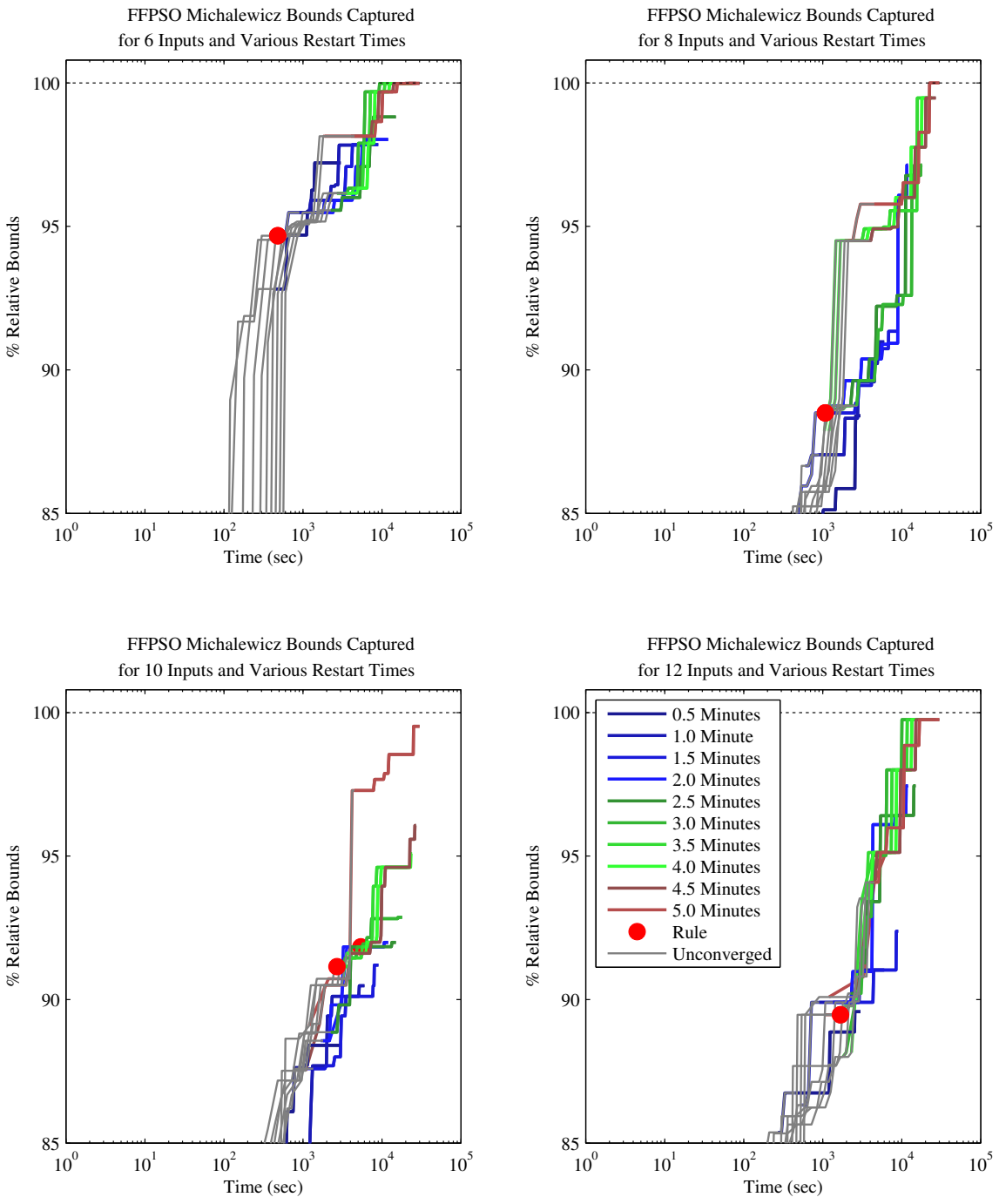


Figure 71: FFPSO Test Function Performance for Michalewicz Function for up to 12 Inputs.

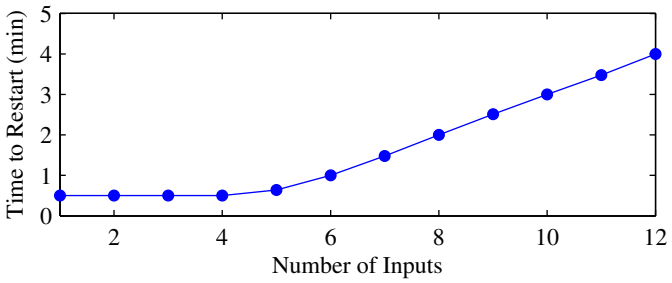


Figure 72: FFPSO Restart Time Relative Convergence Rule vs. Number of Inputs.

Table 5: FFPSO Performance for Schwefel Function with 6 to 12 Inputs.

# Inputs	6	8	10	12
% Relative Bounds	99.22%	98.26%	94.94%	90.47%
Total Time (sec)	900.58	1800.9	2700.9	1440.3
Total Time (min)	15.01	30.02	45.02	24.01
# Samples	15	15	15	6
Minimum σ	0.96	1.41	1.21	1.67

for the Schwefel and Michalewicz functions, given the relative convergence rule provided in Figure 72.

From these two tables one will note that there are no instances where more than 15 samples were taken. This is because an additional criteria was added to the relative convergence rule previously developed. In looking at Figures 70 and 71 cases may require as many as 30 or 50 samples to decrease the standard deviation enough for convergence. When these methods were prematurely terminated after 15 samples, there was little or no loss in the bounds captured. Consider Tables 5 and 6. All cases which terminated with 15 samples performed well when capturing the bounds, with the exception of the 10 input variants, and each of these captured over 90% of the bounds with this premature termination. For this reason, this adaption was applied to the rule.

Table 6: FFPSO Performance for Michalewicz Function with 6 to 12 Inputs.

# Inputs	6	8	10	12
% Relative Bounds	94.68%	88.50%	91.15%	89.47%
Total Time (sec)	480.56	1080.7	2701.7	1680.7
Total Time (min)	8.01	18.01	45.03	28.01
# Samples	8	9	15	7
Minimum σ	1.58	1.84	1.35	1.51

6.4 Using Experiment Results to Quantify Epistemic Model Uncertainty

Throughout this chapter, a series of experiments was conducted to assess the various methods' ability to capture the bounds of the space. The bounds of the space are of interest because they represent the uncertainty in a result. Evidence theory provides a framework for more appropriately making uncertainty statements based on the information available. Given the sparse data available to quantify uncertainty in the Fidelity Selection Problem (FSP), a conservative statement is being made, specifically that only the uncertainty bounds are of interest, rather than the probability of some event occurring. Using the methods identified in this chapter, the uncertainty in a result can be quantified.

Through this chapter, the number of inputs and outputs have been considered. While these may be clear for typical uncertainty studies where the variation in some input parameter is considered, this is not the case for model uncertainty. Model uncertainty is being quantified by applying some "adder" function to the result of deterministic analyses. The bounds of these "adder" distributions are quantified from physical test data that is available for the same class of problem being examined in the uncertainty study. The inputs, as have been frequently discussed in this chapter, are not the inputs to the analysis, but are actually the "adder" distributions. This is illustrated graphically in Figure 73.

In this figure, the "adder" distribution is shown as a uniform distribution with an "E" on it, meaning it is an evidence theory distribution (really any possible distribution). The

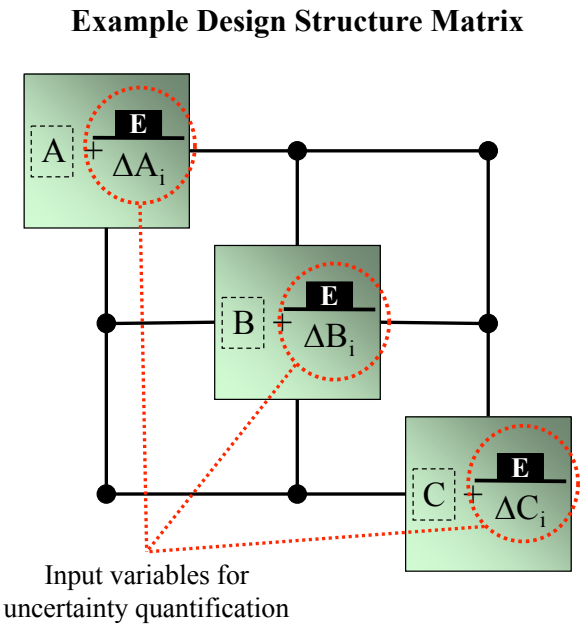


Figure 73: Illustration of the Inputs for Quantifying Model Uncertainty in a Notional DSM.

inputs for a model uncertainty quantification study with this model are ΔA_i, ΔB_i, and ΔC_i, where the subscript accounts for the possibility of multiple outputs from each Contributing Analysis (CA). The bounds of these "adder" distributions identified from physical test data are the bounds of the input variables; they are used to quantify the bounds for any number of outputs of interest from this Design Structure Matrix (DSM).

Based on the information provided in this chapter, one should use the strategic distribution creation method, which was generated by strategically creating a variety of beta distribution combinations using Halton Quasi Monte Carlo (QMC) sampling sequences (discussed in detail in Chapter 4) if there are 4 or fewer input variables for the analysis. If there are more than 4 input variables, then FFPSO should be used to quantify the bounds (discussed earlier in this chapter). The trade-off between the various possible methods is provided graphically in Figure 74. As discussed earlier in this chapter, a trend line was not created for either FFRCGA or FFCACO because these methods performed so poorly relative to FFPSO for the 4 input variable case. While it is possible that some trade-off between FFPSO and FFRCGA could exist for fewer than 12 input variables, this would cause FFRCGA to have a significantly more shallow slope, which is unlikely. It is *extremely*

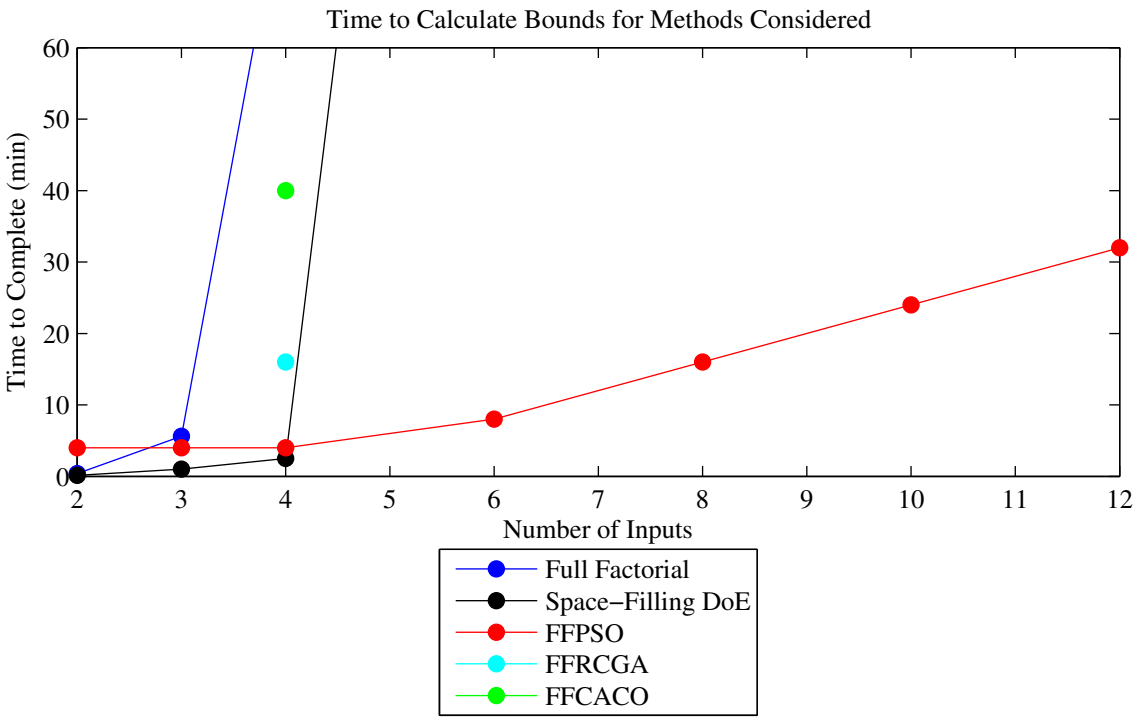


Figure 74: Performance of Considered Uncertainty Quantification Methods vs. Number of Inputs.

unlikely that FFCACO would be a competitive method for fewer than 12 input variables, as this would require it to converge more quickly for higher dimensional problems than for the 4 input case considered. The restart time for the FFPSO should be determined using the trend provided in Figure 72 and this should be run until the relative convergence criteria previously discussed is met.

After the uncertainty analysis has been completed, one combination of analysis codes is represented. In order to solve the FSP, the analysis codes will be "swapped" out (and their corresponding model uncertainty) with another combination. The specifics for this implementation will be discussed in more detail in subsequent chapters.

6.5 Conclusions

In this chapter, a variety of experiments was conducted to quantify epistemic model uncertainty present in analysis codes. Issues with the current implementation were discussed. This led to a series of questions about how one should go about quantifying epistemic model

uncertainty when taking an evidence theory approach. These questions led to a series of experiments which tested hypotheses that answered the questions. While the initial results confirmed the hypotheses, they also created a variety of unforeseen problems. These problems led to additional hypotheses and experiments to test them. These experiments identified two methods for quantifying epistemic model uncertainty, specifically a strategic distribution creation method which should be used if there are 4 or fewer inputs in model uncertainty quantification, and Frontier Finding Particle Swarm Optimization (FFPSO), a modified, multi-objective form of PSO which was originally developed by Kennedy & Eberhart to capture the bounds of the space. Experiments were conducted on how these methods should specifically be implemented and how long they should be allowed to run. The results from these experiments allow one to not only quantify epistemic model uncertainty, but to also use that uncertainty to solve the FSP. The FSP will be addressed in subsequent chapters.

CHAPTER VII

SURVEY OF RELEVANT DISCRETE OPTIMIZATION METHODS FOR SOLVING THE FIDELITY SELECTION PROBLEM

The purpose of this chapter is to begin answering the first discrete optimization research question, specifically **what optimizations method will consistently find the correct answer to the Fidelity Selection Problem?** This chapter will begin by discussing why the Fidelity Selection Problem (FSP) should be considered an optimization problem. Next the chapter will survey relevant optimization literature and discuss possible methods which might be used. Following this discussion, several FSP-specific issues which eliminate many of these options will be discussed. This chapter will conclude with identifying three promising optimization methods which will be implemented in experiment 4 to determine which consistently finds the correct answer to the FSP.

7.1 Formulating Fidelity Selection as an Optimization Problem

Fidelity selection is complex, and because of its importance, which was outlined in Chapter 1, traceability and rigor are needed in fidelity selection. The proposed approach will solve the FSP, as well as add traceability and rigor, by posing it as an optimization problem. However, when solving the FSP, one is not simply optimizing a continuous parameter or even a series of continuous parameters, as is done in most optimization literature. The FSP involves making a series of selections about which code to use to perform each part of the analysis. This fact makes the FSP a discrete one. To further complicate the situation, requirements such as accuracy and run time are placed on outputs of interest from the resulting Design Structure Matrix (DSM). These requirements can be seen as constraints in the FSP. Therefore, the FSP can and should be formulated as a constrained, discrete optimization problem.

7.2 Discussion of Use Cases' Objective Function in Standard Form

Two specific use cases were mentioned in Chapter 1, specifically:

1. Analysis tool selection to provide necessary concept resolution
2. Analysis tool selection when accuracy goals exist for several different outputs from the multi-fidelity Modeling and Simulation (M&S) environment

These use cases should be restated using Multidisciplinary Optimization (MDO) standard form. The second use case would be as follows:

$$\text{Minimize: } t_{cost}$$

$$\text{Subject To: } c_i - g_i \leq 0$$

Where:

- t_{cost} is the time cost for evaluating the DSM
- c_i is an uncertainty (inequality) constraint
- g_i is the uncertainty in output i for the DSM

Before formulating the first use case in standard form, consider the following thought experiment. If one was to consider a DSM with a single Contributing Analysis (CA) and several choices for this CA, a Pareto frontier could be seen when examining time cost and the magnitude of the uncertainty band in a result, as shown in Figure 75.

In this figure, various code options are shown as green circles. The Pareto frontier can be established by finding the desirable direction of change for each parameter (the time cost should be minimized and the magnitude of the uncertainty band should be minimized) and then selecting points that are as close to this area as possible (bottom left corner). The Pareto frontier is shown with a black dotted line. Formulations such as this are used for bi-criteria or multi-criteria optimization.

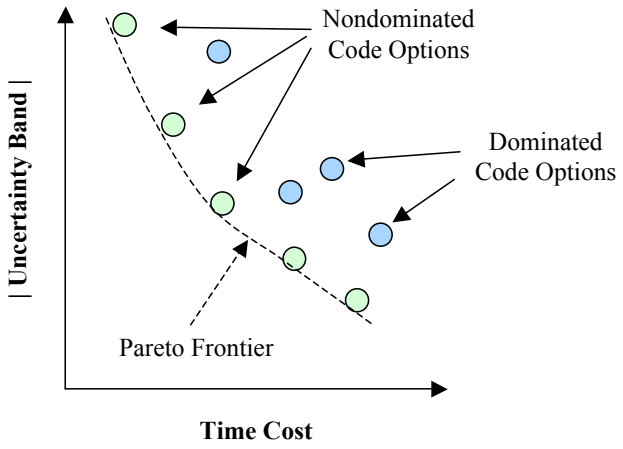


Figure 75: Notional Pareto Frontier for Code Options.

If we now consider this bi-criteria problem in Figure 75 to be a general case, we can limit it by instead applying a constraint in one of the criteria and minimizing (or maximizing) the second. For this example, we have two possible formulations which are shown in Figure 76. The Pareto frontier is shown with a blue dotted line and the constraint is shown in red with the hash marks indicating the infeasible side of the constraint. The first formulation of a constrained optimum is to minimize time cost subject to a maximum uncertainty constraint, and the second is to minimize uncertainty subject to a maximum time cost constraint, which are shown in the left and right, respectively.

The formulation on the right most directly applies to the first use case, and it would require some iteration given the likelihood that different "adders" will exist for each concept. It is interesting to note that the solution obtained for the first formulation (left) with c_1 can be obtained by iteratively solving the second formulation (right) with c_2. This is equivalent to moving the constraint up or down until the appropriate value is found. Also, it is quite possible that finding one formulation may be simpler than finding the other given nonlinearity in the epistemic model uncertainty with respect to the run time of the DSM. Therefore, either formulation could be used, though with differing levels of iteration, to solve the first use case. A selection should be made based on convenience of implementation and the amount of time required to conduct the optimization.

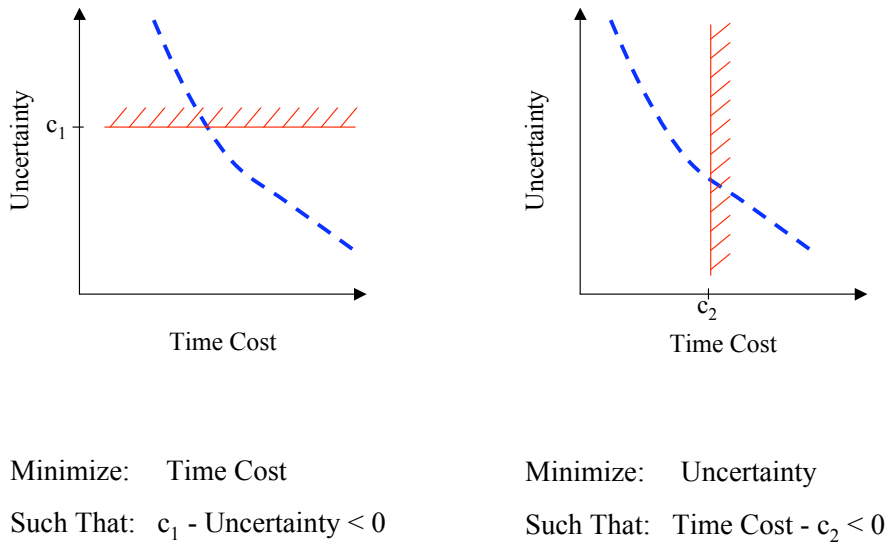


Figure 76: Graphical Comparison of Different Objective Function Formulations.

7.3 Fidelity Selection Recharacterized

Because of the discrete nature of the FSP, it may be advantageous to examine alternative methods for visualizing the DSM being modeled. Traditionally, MDO problems involving linked codes in an integrated Modeling and Simulation (M&S) environment are visualized using a DSM. However, it may be more appropriate to view the FSP using a graph theory approach, specifically representing it as a directed graph or digraph. A graphic illustration of this can be seen in Figure 77. The DSM assumes the codes will be executed moving from the top left corner to the bottom right corner. In a DSM, linkages represent information being passed from one code to another (for example, information from A is passed to B, C, and D). With a DSM, the multi-fidelity aspect is not typically represented at all, but it is shown here with depth (for example, A has two choices).

This MDO approach differs from a graph theory one, but the same information can be displayed. In a digraph, each vertex or node is a code choice (A1, A2, B1, etc). Linkages are used in a digraph to represent possible paths that can be taken. Here, the analysis begins at "Start" and the user must choose one of two options for analysis A, one of three options for analysis B, and so on until reaching the "Stop" node. It is assumed that all information is available when a node is reached. While this is an alternative approach to using a DSM,

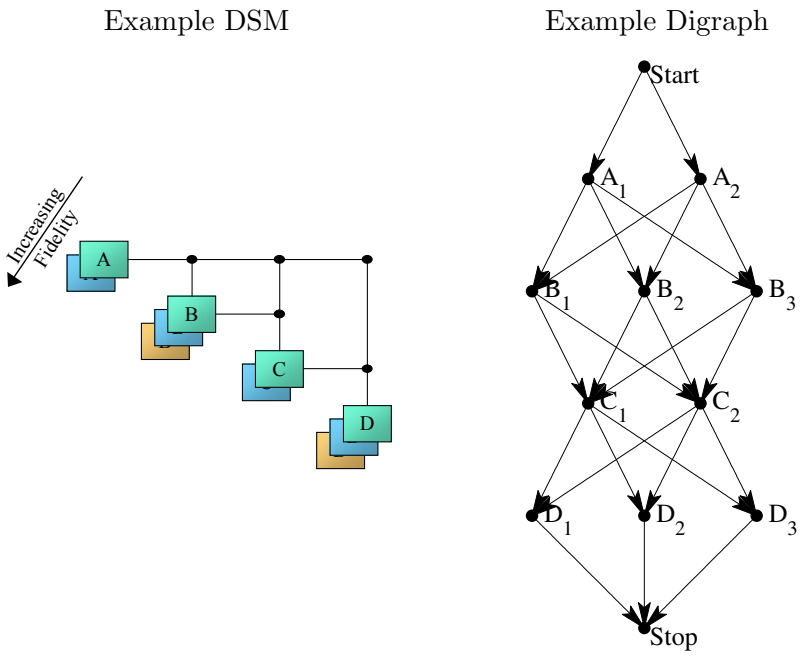


Figure 77: Comparison of an Example DSM and Digraph.

the choices for each analysis can be more clearly represented as well as the need for a traceable, rigorous selection process. It also allows one to show code choice compatibility. Digraphs can be used to visually represent what is occurring in a constrained Shortest Path Problem (SPP) algorithm and are commonly used in Operations Research (OR) and Management Science (MS) applications. Because this visualization lends itself well to the FSP, it will be primarily used in lieu of a DSM.

7.4 Graph Theory and Shortest Path Problem Applicability to Fidelity Selection

In the previous section, graph theory was mentioned as a useful way to visualize the FSP, but little was stated about what graph theory is or how it can be employed. Graph theory is a branch of discrete mathematics which focuses on combinatorial structures of vertices and edges, or nodes and arcs respectively.[83] The basic method for representing information in graph theory is with a graph. A modification of this graph, which is applicable to the FSP, is to imply a direction for travel on an edge. These are referred to as directed graphs or digraphs. This implied direction is commonly shown with arrows rather than lines for edges

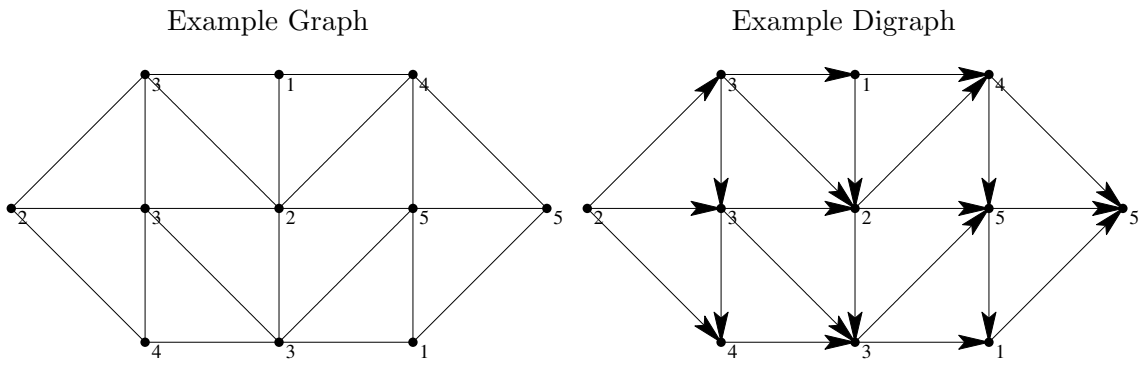


Figure 78: Examples of a Graph and Digraph.

linking vertices. An example of a graph and a digraph are provided in Figure 78. These structures, defined as $G = (V, E)$, can be used to model many things from roads and traffic flow to software programs to economic and social models.[83]

One discrete optimization algorithm used on graphs and digraphs is the SPP. The SPP is commonly implemented on weighted graphs, $G = (V, E, W)$, where weights, W, are assigned to either the vertices, V, and/or edges, E. The algorithm will find the shortest weighted distance from a source vertex, S, to a target vertex (or vertices), T. Stated formally, SPP is the following problem:

$$\text{Given Graph: } G = (V, E) \text{ and weights: } w_{ij} \forall e_{ij}$$

$$\text{Minimize: } \sum x_{ij} w_{ij}$$

$$\text{Where: } x_{ij} = \begin{cases} 1 & \text{if } e_{ij} \text{ is in the path} \\ 0 & \text{otherwise} \end{cases}$$

An illustration of one such graph is provided in Figure 79. Here, the source vertex is the left-most vertex and the target vertex is the right most vertex. Each edge's weight is shown on the edge in black and each vertex is labeled with a letter in red near the vertex.

7.4.1 Common Shortest Path Problem Methods

While the objective of an SPP algorithm is always the same, to find the shortest path from the source to the target, there are a variety of different methods for obtaining this. Three

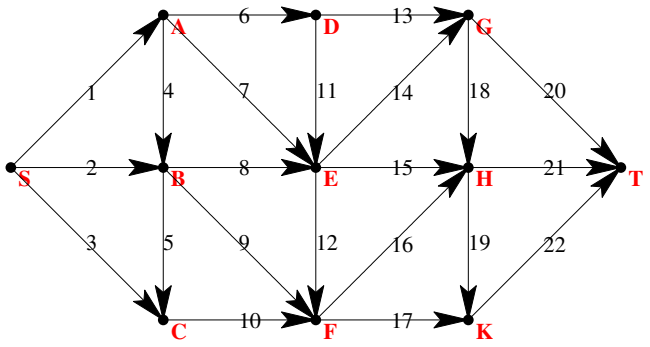


Figure 79: Example Graph for SPP Algorithm.

commonly seen methods are:

1. Label setting schemes
2. Dynamic programming schemes
3. Iterative matrix operation schemes

Each of these three methods will be briefly described and examples of each will be provided when applicable in the following subsections. While this list is representative of popular methods, it is not intended to be an inclusive list of all SPP methods. For a more thorough and detailed taxonomy of SPP methods, both by information available in the graph and the types of graphs being operated upon, please consult Deo and Pang's 1984 article in Networks.[52]

7.4.1.1 Label Setting Shortest Path Problem Schemes

The first method, label setting schemes, generally divides a graph into two subgraphs. An explored subgraph consists of vertices the method has explored (and each has an associated cost of arriving at that vertex) and an unexplored subgraph. The method involves moving through the explored subgraph and moving unexplored vertices which border the explored subgraph into the explored subgraph. If a vertex is reached from multiple edges, the label (cost) assigned to reaching the vertex can be updated with a potentially newer, smaller label (cost). Several popular algorithms operate using a label setting scheme, specifically

Dijkstra's algorithm, which was proposed in 1959[54], A*, ,./ which was proposed by Hart, Nilsson, and Raphael in 1968[93], and D*, which was proposed by Stentz in 1994[202].

Dijkstra's algorithm is a "greedy" algorithm, which will always take the shortest step possible. When referring to Figure 79, Dijkstra's algorithm would first move along edge from S → A. The second step would be along the edge from S → B and the third from S → C. The fourth step would be from A → B, and so on. Comparing to continuous optimization algorithms, this process acts much the same way a steepest descent algorithm would, always taking the shortest step (or step along the steepest gradient) regardless of the direction it might be in.

A*, which was published 9 years later, advanced Dijstra's original algorithm by adding heuristics to it. Rather than always taking the shortest path, A* incorporates heuristics if additional information is known. For example, if one were trying to drive from Atlanta, GA to Washington, D.C., one would expect to proceed northeast until reaching the destination. When examining a graph of the trip, the source may contain two edges, one edge leading to Greenville, SC (North on I-85) and the other edge leading to Macon, GA (South on I-75), Dijstra's algorithm would first go to Macon, GA because the cost (distance) to travel there is lower. A* can be configured with heuristics which would instead use additional information, such as geographic location when selecting the direction. Here, A* would instead make the correct choice and travel to Greenville, SC instead of in the opposite direction on a shorter edge to Macon, GA. Incorporating heuristics such as geographic location in SPP can be paralleled to continuous optimization algorithms such as Fletcher-Reeves. Here, this algorithm not only calculates the local gradient and uses it to guide the algorithm's path, but it also considers where its previous steps have been to try to more efficiently move.

The third method mentioned, D*, was named based on its similarity to A*, but with one adaptation: D* is used on graphs with incomplete information and dynamically updates graph weights and edges. This was done because in Artificial Intelligence (AI) applications, such as autonomous rovers on other planets, the topography is not entirely known when proceeding to a distant target. For example, if a rover was attempting to move from one

hill to another, it would choose the shortest path, a direct one. If, while moving toward this target hill, the rover were to become aware of a crevice not previously visible, D* would allow the rover to re-assess its environment and plan a new course it could traverse.

7.4.1.2 Dynamic Programming Schemes

A second commonly used approach for solving SPP is with dynamic programming. While this may seem to be a specific method, it is actually a "general type of approach to problem solving, and the particular equations used are developed to fit each individual solution"[105], much like label setting schemes. Rather than branching out and searching each vertex, and possibly replacing the current label with a better one, dynamic programming takes a different approach. It begins by solving a small subproblem of the actual one. This subproblem recursively increases in size and is solved based on the previous subproblem solution until the entire problem is solved.

According to Hillier's text, there are six basic properties of all dynamic programming problems:[105]

1. The problem is divided into subproblems where a decision is required at each.
2. Each subproblem has a number of possible choices associated with it.
3. The procedure will solve for the optimum solution to each subproblem.
4. At a given subproblem, the solution must be independent of every other subproblem.
5. The system is solved backwards, beginning at the end and working towards the beginning.
6. A recursive relationship that identifies the optimal solution for the current subproblem, given the solutions to the previous subproblems is available.

While this may seem somewhat confusing, an illustration with an example digraph for the FSP may be helpful. This is provided in Figure 80.

This illustration is of the original digraph presented in Figure 77, but transposed; the direction of the edges has been flipped for a dynamic programming problem, and the "Start"

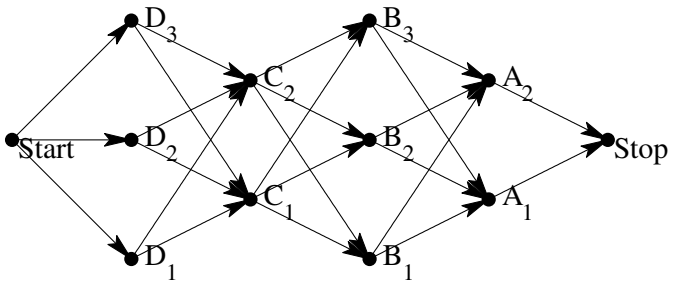


Figure 80: Example Graph for Dynamic Programming.

and "Stop" have been switched. If the FSP is broken into several subproblems, each subproblem would be selecting a given tool for each CA in the DSM. First, the algorithm would find the "optimum" (least costly) solution for that subproblem. Next, it would solve the "optimum" subproblem of determining which tool should be chosen for CA C, given the choice of CA D. This would continue, moving "up" the DSM as the algorithm moves from left to right in Figure 80 until it finally reaches the "Stop" (which was previously the "Start" for the DSM). This process was initially proposed by Richard Bellman in his 1957 book[18] and further outlined in his 1958 paper on the subject[19].

Dynamic programming principles have been used for a variety of different applications, including, but not limited to, shortest path problems.[105] Some have used it to not only solve unconstrained problems as Bellman initially formulated the method, but also for constrained path problems [114, 186]. As stated earlier, dynamic programming is more of a highly adaptive mindset used to solve problems, unlike label-setting schemes which have a series of specific algorithms. An analog to this in dynamic programming would be Bellman's equation. This is the equation he solidified in his 1957 book on dynamic programming, and is simply the 6^{th} dynamic programming property stated as an equation.

7.4.1.3 Iterative Matrix Operation Schemes

Iterative matrix operations are a third common method used in academic literature to solve SPP. This formulation first involves constructing an adjacency matrix, A, for the graph, $G = (V, E, W)$ of interest, where A_{ij} is 1 if the edge $v_i v_j$ is in G, otherwise, it is 0. This matrix is then added to its transpose repeatedly, and when A_{st} becomes nonzero, the

shortest path from source s to target t has been identified.[91] While this method can be clearly explained and understood, it is not very computationally efficient. For this reason, it is not commonly used except in an academic setting to explain how SPP works.

7.5 Constrained Shortest Path Problem Methods

Now that general, unconstrained methods for calculating a shortest path have been identified, a discussion of constrained SPP is in order. This taxonomy was identified initially by Deo & Pang[52] and further populated by Beasley & Christofides in their 1989 Networks article.[17] Since this article, many others have used this taxonomy to describe constrained SPP[77, 148, 209]. For the purposes of this document, we will use this taxonomy, which divides constrained SPP into three separate categories: Resource Constrained Shortest Path (RCSP), Vertex Constrained Shortest Path (VCSP), and Time Constrained Shortest Path (TCSP).

The analogy commonly used to describe SPP is of a person traveling from one location to another. This can be further used to describe these three cases. RCSP would be the person traveling from one place to another, but with a limited amount of funding (resources). This could perhaps be renamed the "broke college student on spring break" problem. The second case, VCSP, is similar to RCSP, but instead of limiting the traveller on the amount of resources they have, they are limited on the number of locations they can stop at along the way (vertices). While this could really be thought of as a special case for the RCSP, the distinction will be made here to maintain consistency with literature on constrained SPP. The third case, TCSP, could be thought of as a person going from one place to another, but they must pass through certain locations at certain times. This type of formulation is useful for delivery systems but not for the FSP. The FSP is an RCSP, and therefore this will be the only type of constrained SPP considered from this point forward. More formally, the RCSP can be stated as:

$$\text{Given Graph: } G = (V, E) \text{ and weights: } w_{ij} \forall e_{ij}$$

$$\text{Minimize: } \sum c x_{ij} w_{ij}$$

$$\text{Subject To: } \sum x_{ij} \vec{r_{ij}} - \vec{R} \leq 0$$

$$\text{Where: } x_{ij} = \begin{cases} 1 & \text{if } e_{ij} \text{ is in the path} \\ 0 & \text{otherwise} \end{cases}$$

$$r_{ij} = \text{resource(s) needed for edge } e_{ij}$$

7.5.1 Popular Resource Constrained Shortest Path Methods

SPP methods discussed in the previous section were all fairly similar in that they operated by taking the shortest step or something involving a shortest step to arrive at the solution. This occurs because SPP is a Polynomial (P) time algorithm. While P time is a complexity theory term, it could be paralleled to modality in a system from an optimization perspective. Algorithms for P time problems are solved given a non-exponentially growing amount of time using gradients or "gradients" (probe steps) because the problem being solved is unimodal. Unimodal algorithms in optimization will always arrive at the same optimum (neglecting round-off error) regardless of the starting point. Physically, this corresponds to attempting to find the bottom of a bowl. Regardless of where the algorithm starts, a P time (or gradient-based) method will move to the same solution.

Moving on to RCSP, the problem has (unfortunately) been proven to be a Nondeterministic Polynomial (NP) problem.[8, 148]. A common mistake is that NP actually stands for "not polynomial".[171] While these problems are indeed not polynomial, it actually refers to the nondeterministic methods commonly used to solve them.[41] This essentially means that when an algorithm reaches a crossroads and must decide what to do, taking the gradient (or "gradient" as is the case in SPP) will not necessarily guide the method to the global optimum. Rethinking this in terms of the system modality, NP problems are equivalent to multi-modal problems. One common 2-D example is the egg-crate problem (envision an egg-crate tilted on one corner). Depending on the starting condition, gradient-based

algorithms tend to get "stuck" in a local minima and are unable to "jump" to another spot in an effort to find a better solution. Therefore, while the SPP methods were generally entirely "gradient" based, RCSP methods will have a "gradient" aspect but must also have a non-gradient aspect, allowing them to "jump" to other regions of the graph and find potentially better solutions.

Four common types of methods have been identified in the literature, specifically K-shortest paths, cutting plane Linear Programming (LP)-based methods, dual programming problems, and dynamic programming methods. Each will be discussed and illustrated. Following this, issues with certain methods will be identified. Research questions will be asked about these issues specific to a FSP. After a literature review and qualitative comparison of these various methods, hypotheses to these questions will be posed.

7.5.1.1 K-Shortest Paths

The K-shortest paths method for solving RCSP actually involves using the SPP K-shortest paths method. Rather than simply finding the shortest path from a source to a target, K-shortest paths finds the shortest, $2^{\text{nd}}, 3^{\text{rd}}, \ldots, K^{\text{th}}$ shortest paths from the source to the target. The idea of using K-shortest paths to solve a problem was initially pursued as a "poor man's" way of examining multiple promising solutions without imposing constraints. For example, rather than explicitly stating constraints, one could instead examine the best few solutions to a problem and then select based on "expert opinion" or other criteria which cannot be easily stated explicitly.[193]

When examining K-shortest paths, there are actually two different classes of algorithms. The first is a class of methods which do not allow solutions to pass through a given vertex more than once. The second is a class of methods which do allow solutions to cross. The first class includes methods developed by Yen in his 1971 paper on loopless K-shortest paths[229]. This also include methods developed by Lawler in 1972[129], among others. The second class of algorithms, where repeated vertices are allowed, was formally proposed by Shier[193], though the basic framework for this sort of algorithm can be traced back further to more introductory texts[91]. Because the FSP involves systems where vertices

cannot be repeated (A will not be called after B unless a feedback, which would be required, not optional, exists), this class of methods will not be considered and only non-repeating vertex shortest path methods will be examined.

Non-repeating vertex shortest path methods can be further broken down into those which allow loops and those which do not (a loop in graph $G = (V, E)$ is an edge which begins and ends at the same vertex). While it is true that any loop in the digraph of a DSM would be nonsensical, these algorithms should not necessarily be excluded from consideration. These methods are discussed in detail in Yen's 1971 article [229] and Lawler's 1976 book [130]. Yen's method is outlined here, but one should note that Lawler's method is essentially the same.

Good properties of the K-shortest paths method for finding the RCSP is that the method is relatively easy to implement and understand. Additionally, one is guaranteed to eventually find the solution. The drawback that the convergence rate for finding an optimum RCSP is dependent on how "constraining" the constraint is and how large G is. If 70% of the possible combinations of paths are infeasible, the algorithm must first traverse these to find the optimum. Additionally, if G is large, the inner-most iteration in Yen's K-shortest paths will require a significant amount of time. The procedure for finding the K-shortest path is provided in Listing 7.1.

7.5.1.2 Cutting Plane Linear Programming

One common method for optimization in general as well as for MS and OR research is LP. LP is a constrained optimization technique which works efficiently for linear objectives with linear constraints of the form $y = Ax + b$.[212] Because of the assumed linearity of the system, LP programs assume all inequality constraints are equality constraints, and move along the bounds of the feasible space to find an optimum solution.

Formally, LP problems are formulated as follows:

Listing 7.1: K-shortest Paths Calculation

```
function[AK] = KshortestPath(G,S,T,K)
%G = G(V,E)
%S = source node
%T = target node
%K = Kth shortest path being found
[A1,c1]= SPP(G,S,T) %find shortest path in G
%A1 is shortest path
%c1 is cost for path A1
Q=size(A1) % A1 has Q vertices
for k=2 to K
    Atemp=[ ]
    ctemp=[ ]
    for i=1 to Qk
        Gtemp = G without vertex i from Ak
        [Ai,ci]=SPP(Gtemp,S,T)  %find shortest path in Gtemp
        ctemp=min(ci,ctemp) %cost for Ai is lowest cost found so far
        if ci = ctemp
            Atemp=Ai %save shortest path if it is lowest cost found so far
        end
    end
    AK=Atemp
    cK=ctemp
end
```

$$\text{Minimize: } F(x) = \sum_{j=1}^{n} x_j c_j$$

$$\text{Subject To: } \sum_{j=1}^{n} a_{ij} x_j - b_i \leq 0 \text{ for } i = 1, 2, \ldots, m$$

$$x_j \geq 0, \text{ for } j = 1, 2, \ldots, n$$

LP is commonly done using what is referred to as the Simplex method.[45, 212] As one can see, x will increase in size for each additional dimension to the problem, and b will increase in dimension for each constraint added to the problem. The Simplex algorithm performs matrix operations where the efficiency is a function of the number of constraints, meaning the fewer constraints a problem has, the more efficiently it will solve the problem. This is useful because in many LP formulations, there are generally only a few constraints but the number of variables may be quite large. This Simplex algorithm was initially proposed by Dantzig in 1947 as a way to solve LP problems.[105, 219] Dantzig and Wolfe went

on in their 1960 paper to discuss a special and popular case for Simplex, the decomposition principle, which generalizes linear programming and the Simplex method.[46]. This approach has been used by many to solve problems[73, 140], and it will be discussed in more detail later.

The Simplex algorithm has been used and added to by many others. One relevent special case for the Simplex LP problem is Integer Programming (IP). Here, an additional series of constraints are placed on the system, specifically that all x_i must be integers.[80, 81] A further special case of IP is Binary Integer Programming (BIP), a problem where not only must the solutions be integers, but they must be either a 0 or a 1. A common generalization of IP used for discrete problems is the branch and bound method initially proposed by Land & Doig in 1960.[128] In their article, the authors proposed a system where a discrete system could be formulated as a further constrained LP problem.[175]

Branch and bound consists of three phases: branching, bounding, and fathoming.[105] Before any of these can begin, the algorithm first solves the continuous problem, ignoring any discrete constraints. Next, the problem is branched and the discrete variable furthest from a discrete setting has additional constraints, or bounds, placed on it. These bounds are constraints which require the solution to be less than or equal to the lower feasible bound (setting) or else greater than or equal to the upper feasible bound (setting). The solutions to these LP subproblems are compared and the better of the two is kept. The problem is then branched on the next discrete variable that is not at a discrete setting. The problem is bounded to the upper and lower settings of that variable. If, when evaluating these two branches for the second variable, we find that the less desirable solution from the first branch (which was not chosen) is less desirable than either of these new bounded solutions, the first undesirable branch is fathomed, or removed from further consideration. The fathoming process could be thought of as pruning this "tree" created from branching subproblems.

An illustration of this process may be helpful. Consider posing the following question to a student in a class: "where is the best place to sit during class?". There are 2 inputs, the row and column (or seat on that row), which could be chosen. One should note that the

question being asked can have a continuous solution, but the student is actually constrained to physically sitting in a seat. The mysterious objective function is first assessed assuming the student isn't constrained to actually sit in the seat (continuous problem). After assessing this mysterious objective function, the experimenter is told the best place to sit is in the 2.4^{th} row, in the 13.15^{th} column. Clearly this is a nonsensical answer, but it could be considered a starting point. Next, the solution is branched (perhaps because the school's janitorial staff would be quite unhappy with students standing on the desks given the need for more thorough cleaning), and the student is asked the same question, but bounded, in that they can either sit in the 2^{nd} row or closer to the front, or in the 3^{rd} row or further back. This imposes a discrete constraint on the row the student can sit in with respect to the previous optimum. After considering this new constraint, the student says the best place to sit is in the 3^{rd} row, and in the 12.2^{th} column. Next, the column variable is considered (branched) and an additional constraint is now placed on problem and the student is (bounded) told they can sit in either the 12^{th} column or further to the left or in the 13^{th} column or further to the right. After thinking the problem over, the student says the best place to sit is in the 3^{rd} row, 13^{th} column, which corresponds to a seat.

This method is also referred to as a "cutting plane" LP problem, because in 2-D, these additional bounds represent constraints, or planes that cut the feasible space into sections. Although confusing, it would perhaps be better to refer to these problems as cutting hyper-planes rather than cutting planes unless referring to a problem in 2-D. IP and BIP schemes commonly enforce their integer and binary constraints by using cutting planes. Schemes have been used to solve graph problems by formulating them as a BIP problem where each vertex is an input variable and $x_i = 1$ if v_i is in the path and is 0 otherwise.[17] One should note that, as stated earlier, the efficiency of the Simplex algorithm is a function of the number of constraints, and it requires more computation to solve a problem with a large number of constraints compared to a smaller number of constraints. As stated earlier, IP problems are formulated by adding additional constraints to the problem, specifically 2 per integer setting. As the number of integer variables increases, the number of constraints quickly grows. Similarly, when a BIP approach is taken to solve RCSP problems, it becomes quite

costly.

7.5.1.3 Dual Problem

When formulating LP problems, one important discovery was the presence of a dual problem, which has since led to duality theory. Duality theory states that for every original (or primal) problem, there also exists a dual problem. This dual problem has been used for sensitivity analysis, including shadow prices in stocks and management decisions, among other things[105]. To describe the dual problem, let us first consider the original, or primal problem, in standard form.

$$\begin{aligned} \text{Minimize: } & F(x) = \sum_{j=1}^{n} x_j c_j \\ \text{Subject To: } & \sum_{j=1}^{n} a_{ij} x_j - b_i \leq 0 \text{ for } i = 1, 2, \ldots, m \\ & x_j \geq 0, \text{ for } j = 1, 2, \ldots, n \end{aligned}$$

If we now consider the dual problem, we can see similarities to the primal.

$$\begin{aligned} \text{Maximize: } & y_0 = \sum_{i=1}^{m} b_i y_i \\ \text{Subject To: } & \sum_{i=1}^{m} a_{ij} y_i - b_i \geq 0 \text{ for } j = 1, 2, \ldots, n \\ & y_i \geq 0, \text{ for } i = 1, 2, \ldots, m \end{aligned}$$

In formulating the dual problem, we have added two additional vectors, y and y_0. The y vector corresponds to Lagrangian multipliers, and the y_0 corresponds to the shadow price, or the price/value that the solution must change to for the optimal solution to change, meaning a different set of constraints would be active.

The primal-dual relationship can perhaps be seen more clearly when displayed graphically as a table in Figure 81. It should be noted this was also done by Hillier in his text.[105] From this table, we can essentially see that the dual problem is solving the transpose of the A matrix, or the matrix of coefficients from the primal problem.

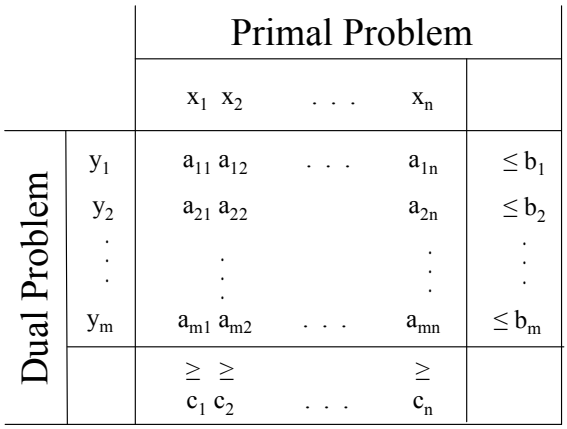


Figure 81: Graphic Representation of Primal and Dual Problems.

Once the dual problem for a linear problem has been formulated, the problem can be solved using a dual Simplex algorithm. This is the same as the Simplex discussed earlier, but the dual problem must be transformed into an appropriate form for the Simplex algorithm. It should be noted that because the problem has been transposed, it can now be solved more quickly. For this reason, dual approaches to integer problems are generally preferred to primal ones.

When comparing solutions from the primal and dual problems, the weak duality property or theorem states that if $\vec{x}$ is a feasible solution to the primal problem, then $\vec{y}$ is a feasible solution to the dual problem, and therefore we have Equation 27.

$$c\vec{x} \leq \vec{y}b \tag{27}$$

Additionally, if $\vec{x}$ is the optimal solution to the primal problem, then $\vec{y}$ is the optimal solution to the dual problem. This relationship is given in Equation 28.

$$c\vec{x} = \vec{y}b \tag{28}$$

This is important, because when solving a unimodal problem, we can always apply the strong duality property. If, however, we are solving a multi-modal problem such as RCSP, only the weak duality theorem can be applied.

As was stated earlier, the dual of a problem could be thought of in (admittedly) simplistic

terms as solving the transpose of the original problem. Recalling the discussion at the conclusion of the previous section, the issue with implementing IP for solving RCSP is that it must be recast as the same problem but with a large number of constraints. This made the Simplex algorithm extremely inefficient, though it can be used. If, however, one were to formulate the dual of this RCSP problem, they could much more efficiently solve the problem. This is the more common approach to solving RCSP because of this fact.

The first and most common implementation of duality to solve RCSP was done by Handler & Zang in their 1980 Networks article for a single constraint.[90] While the algorithm seems quite straight-forward, implementing duality and then solving the system, one should remember the requirements for comparing primal and dual solutions. The strong duality theorem, which would be preferable in this case, can only be applied if the solution is an optimal one. However, as stated earlier, RCSP is a multi-modal problem. Therefore, this may not be the case and Handler & Zang were only able to apply the weak duality theorem. This leads to the potential presence of a gap between the feasible and optimal dual solutions, which is called the "duality gap".

The process implemented by Handler & Zang can be described with 3 phases:

Phase I: Initialization

1. Solve for the unconstrained SPP where the objective is to minimize the objective function (upper bound). If this meets the constraints, then SPP meets RCSP; then stop. otherwise...

2. Solve for the unconstrained SPP where the objective is to minimize the constraint function (lower bound). If the solution exceeds the constraints, RCSP has no solution; then stop. otherwise, go to Phase II.

Phase II: Solving the Dual Problem

1. Plot upper and lower bounds' dual solution as a function of y_0 vs. y, where y is the Langrangian multiplier and y_0 is the shadow price.

2. Use the intersection of upper and lower bounds as the new candidate optimum, then calculate SPP for this new dual problem.

3. If the new solution exceeds the constraints (infeasible) but less than the upper bound, use the new candidate to replace the upper bound. If the solution meets the constraints (feasible) by less than the lower bound, use the new candidate to replace the lower bound.

4. Repeat steps 2 and 3 until finite upper and lower bounds have been created, then go to Phase III.

Phase III: Closing the Duality Gap

1. Evaluate next K-shortest path.

2. If the constraints are met, or if the lower bound is greater than the upper bound, terminate process, otherwise ...

3. If the constraints are not met:

   3.1. If K-shortest path's y_0 is better than the lower bound, replace the lower bound with y_0, then restart Phase III.

   3.2. Restart Phase III.

As stated earlier, this formulation by Handler & Zang was for a single constraint, but they stated that the basic formulation would hold for multiple constraints. This Handler & Zang formulation has been used by many to solve RCSP problems.[148, 194] The process was later improved by Beasley & Christofides, who proposed adding a phase between Phase II and Phase III for Handler & Zang's method; this essentially prunes the graph, eliminating any paths which can obviously be excluded as candidates for the RCSP. This accelerates Phase III of the original method.[17]

The advantage to using this approach is that in formulating the dual problem, the IP problem with a Simplex-based approach to solving RCSP can be significantly accelerated. However, in doing this, the dual problem must be solved and non-gradient methods are required to find new candidate paths.

The K-shortest paths component of Phase III can be replaced with other non-gradient based methods, including Lagrangian sub-gradients.[17] The Lagrangian sub-gradients method,

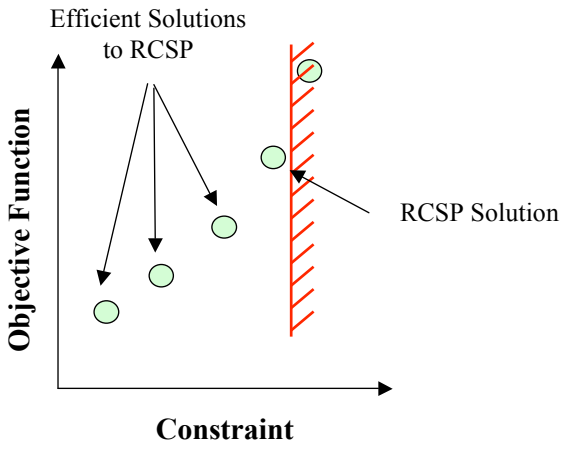


Figure 82: Notional Results from Joksch's Dynamic Programming RCSP Formulation.

like K-shortest paths, is a non-gradient-based procedure for solving optimization problems.[21, 72, 71, 99]

7.5.1.4 Dynamic Programming

Dynamic programming, which was discussed earlier, has also been used by some to solve RCSP.[114, 8, 186] The overall approach for solving the RCSP is very similar to solving the SPP, but with one difference. In the SPP, the Bellman's equation determines the optimal subproblem and then recursively builds the solution from the end of the problem backwards to the beginning.[19] Joksch proposed using a similar approach, but instead of solving for the optimum subproblem, one should find the *efficient* subproblem.[114] By *efficient*, it means a Pareto efficient series of solutions, where one finds the solutions whose performance cannot be improved without sacrificing cost (in this case the constraint value). This method, like dynamic programming in general, is exhaustive, meaning the entire subgraph is evaluated repeatedly. The result from Joksch's method could be notionally illustrated in Figure 82.

One interesting piece of information is that Joksch states in his formulation that when one is only interested in the RCSP solution, as shown in Figure 82, this is not a very efficient method to employ because of its computationally expensive, recursive nature. However, if one were not only interested in the RCSP solution, but also the trade-off surface which results (all efficient RCSP solutions), which is also clearly produced by this method, this

would be an extremely valuable method. This is significant, because while the use cases formulated for this process were only for the RCSP, the trade-off surface may also be important and useful in illustrating the solution that was chosen.

This initial formulation by Joksch was done for a single constraint. It could in theory be applied for multiple constraints, but there is difficulty in visualizing and determining efficient solutions when multiple dimensions (constraints) are involved. Additionally, one should note that as the number of constraints (efficient frontiers) is increased, the number of points on frontiers increases. Given a simple thought experiment, which is left to the reader, one could show that for n solutions and n efficient frontiers, every point is on an efficient frontier, and this would, in essence, defeat the purpose of having efficient frontiers.

Saigal took a slightly different approach by formulating an VCSP procedure, which can in essence be considered a special case for the RCSP problem.[186] While this is of little use for the FSP, it has been mentioned for completeness. Aneja et. al. also proposed a dynamic programming formulation for RCSP, but no computational results were included for their method.[8]

7.6 Issue with Linear Programming and Integer Programming Formulation of Fidelity Selection Problem

Throughout this chapter, SPP methods and algorithms have been taxonomically outlined with a specific focus on RCSP methods thought to be useful for a FSP. These favorable RCSP methods can all be referred to as "exact" methods because when they converge, they produce the exact answer. These exact methods have been either brute-force based (K-shortest paths), Simplex based (cutting plane Simplex, dual problem), or dynamic programming based. There is, however, one issue with Simplex-based and dynamic programming based approaches. Each of these assume the problem is additive. Unfortunately, this is not the case for the FSP. As initially formulated, the objective function for a FSP is the run time for the analysis, which should be minimized. The side constraint(s) are posed as uncertainty requirements in certain outputs. Epistemic model uncertainty is non-additive with the exception of the epistemic model uncertainty in the final Contributing Analysis (CA). For clarification, please revisit Figure 10 in Chapter 2.

Recall that the modification to K-shortest paths to solve a non-additive problem is trivial, so this method can still be applied. The dual problem formulation can be used if an alternative formulation for non-additive paths exists. Dynamic programming, however, should not be considered any longer. The benefit to a dynamic programming approach comes from its ability to recursively solve subproblems and combine the results. If the paths are no longer additive, these paths cannot be combined and the method would actually require more time to evaluate than a completely exhaustive approach.

7.6.1 Non-additive Path Weights Challenge

A graph consisting of only additive shortest paths is an assumption made in SPP and RCSP algorithms discussed up until this point in this chapter. However, as established in the previous section, the assumption of additive paths is not valid for the FSP. Surveying literature, Hillier notes that if the paths for a problem are non-additive then neither Simplex nor dynamic programming can be employed, as they are both linear methods; instead, Nonlinear Programming (NLP) methods should be examined.[105] This is unfortunate, as many methods discussed and published in OR and MS journals are related to traffic flow problems and routing problems, which are linear and mostly additive. There is a limited amount of literature available on Non-additive Shortest Paths (NASPs). Here, the linearity assumption is removed from this method. There are two approaches to handling NASPs:

1. Perform a variable substitution or other manipulation to linearize the problem
2. Use NLP methods.

If one were to take the first option, the linear problem could be solved with many of the OR methods previously outlined. This is the more common approach taken in literature for vehicle routing and traffic equilibrium problems.[76, 77, 209] The second approach is to take a NLP approach. This approach is more involving and requires a more involved approach because of the nonlinearity. Many classic sources state that a nonlinear approach is too expensive and should not be pursued [212], but this is clearly not the opinion of many who have taken this route in NLP literature.[4, 12, 125] With respect to the FSP,

the first option is not a choice. The function's nonlinearity is governed by the uncertainty propagation, and as established in Chapters 4 and 6, it is likely nonlinear and is clearly problem-specific. Therefore, the second option is the only that can be pursued.

One additional point which is of interest for the FSP and NASP is from the Tsaggouris & Zaroliagis paper previously mentioned.[209] Here, the authors forward the Handler & Zang approach of solving the dual problem and then closing the duality gap, but they modify the method. Rather than explicitly following the method like many have done before or adding steps to the method[17], they treat it as a *framework* rather than a series of steps. This approach seems very promising and will be pursued as a promising candidate for solving the fidelity selection with the following framework:

1. Initialize Problem
2. Solve Dual Problem
3. Prune Digraph, Removing Infeasible Edges
4. Close Duality Gap

The first step of this analysis can remain the same, as it is not significantly impacted by a nonlinear approach. The second step, solving the dual problem, will change because the Simplex algorithm, which only applies to linear, additive problems, cannot be employed. Here, nonlinear dual programming can be used. The graph can be pruned just as outlined by Beasley & Christofides.[17] The final step, closing the duality gap, must also be investigated to find feasible methods.

7.6.1.1 Nonlinear Programming for Solving the Fidelity Selection Problem

As discussed in the previous section, NLP can be used to solve RCSP and the fidelity problem using a Handler & Zang framework. Here, the applicability of NLP will be made and the specific type of NLP that could be used will be identified based on the results of a literature review.

The bulk of NLP methods, as with most optimization methods, are developed for continuous problems. However, as established earlier, the FSP is a discrete constrained optimization problem. Therefore, popular methods such as Sequential Quadratic Programming (SQP) cannot be used. Nearly all NLP methods require a problem to be differentiable, and many require second differentials to solve, including SQP.[21, 212]. Some methods are able to linearize the space and operate by only using first order derivatives, but regrettably the FSP, or RCSP with non-additive paths, can be formulated as an Integer Nonlinear Programming (INLP) problem, where a vertex receives a 1 if in the shortest path and a 0 if it is not. By using such a formulation for a nondifferentiable problem, there are two nonlinear approaches which can be examined: a cutting plane NLP approach and that of Lagrangian sub-gradients.

A cutting plane method is very much like the linear cutting plane formulation. The difference is that the constraints are more costly and difficult to impose and the method does not have a neat Simplex formulation. However, it could be formulated, although it may be costly to implement due to its nonlinearity.[212] This has, however, been shown by several to have moderate convergence properties for certain classes of problems with linear and nonlinear constraints and objectives.[4] It should also be noted that a dual problem exists in nonlinear programming just as it does in linear programming. The dual NLP problem has received much attention in mathematical programming circles and solutions to general problems have been found with success.[12, 154, 225] It should be noted that while this approach could likely be used to solve a FSP, the difficulty in formulating such an approach would be extreme. Additionally, little research with respect to solving RCSP related problems is being published in the literature for reasons which will be outlined later.

The second nondifferentiable method examined for solving the NLP problem is to use Lagrangian sub-gradients.[21] While the Lagrangian sub-gradients is strictly a nondifferentiable method, it approximates derivatives by examining vertex solutions, or small steps in the same neighborhood. For linear problems, this was an acceptable formulation. It would step in a given direction and essentially assemble a piece-wise linear function. This, however, cannot be done for a nonlinear graph problem but would apply for a general NLP

problem, where derivatives or probe steps (extremely small steps in a given direction) could be taken. For this reason, the Lagrangian sub-gradients method is not a suitable method for solving the NLP RCSP problem.

7.7 Metaheuristic Methods for Solving the Resource Constrained Shortest Path Problem

Metaheuristic methods, which are often also stochastic MDO methods, can be used for discrete optimization problems. Six commonly used methods, specifically Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Firefly Algorithm (FA), and Simulated Annealing (SA), were discussed in Chapter 5. Many of these methods, such as GA, ACO, ABC, and SA, were initially formulated with either mixed or discrete variable problems in mind[227]; therefore, their implementation to solve the FSP would likely be relatively straight-forward. Constraint handling is an ongoing area of research in metaheuristic algorithm development. A variety of schemes, the most popular of which are rejection and penalty functions, were discussed in detail earlier in Chapter 5. Based on the discussion in Chapter 5, only three of these six methods will be considered any further (GA, ACO, and PSO).

7.7.1 Metaheuristic Method Constraint Handling

The three methods discussed earlier, the GA, ACO, and PSO, were discussed in great detail when implementing them to find the frontiers of the space. This subsection will serve to discuss how contraints might be implemented for each of the algorithms. This was not previously discussed as it is not relevant when finding the bounds of a space, but it critical when solving the RCSP.

7.7.1.1 Genetic Algorithm Constraint Handling

Some have shown that modifying an GA to handle constrained optimization is not straightforward, and alternative approaches have been taken, including "multiple island" schemes where various parameters are individually optimized and then combined after the fact. [26] Recently, however, some have shown that success can be met when using constraints for metaheuristic methods, including GAs. The following taxonomy has been proposed for

various constraint methods.[33, 35, 153] These methods have been documented throughout the literature and a quick summary will be provided here for completeness.

1. Methods based on rejection of infeasible solutions
2. Methods based on penalty functions
3. Methods based on multi-objective optimization extensions
4. Methods based on repair algorithms
5. Methods based on specialized operators
6. Methods based on behavioral memory

The first of these methods, the rejection of infeasible samples, is also the most popular, and in some texts is referred to as the "death penalty" function.[35, 153] It has been documented throughout the literature for a variety of problems[49, 50] and could be thought of as an extension of elitism in evolutionary algorithms. Elitism is a common practice in GAs, and in essence, this rejects half of the population of each generation to ensure that the best solutions are maintained. This has been used to solve problems, including the economic dispatch problem[119] and Deoxyribonucleic Acid (DNA) sequencing[131], to name a few. Its popularity is largely due to its ease of implementation, its compatibility with a GA's operations, and its universality. A rejection sampling approach is not problem dependent, and if one were to use a tournament selection method with an elitist algorithm, selection could be easily made where one should maximize fitness, and if both candidates are infeasible, minimize constraint violation. The only requirement is that one must be able to evaluate the infeasible candidates in order to determine which are more fit than others. This approach extends to multi-objective problems as well, though some sort of ranking is required to determine how to choose one infeasible solution over another. One issue with a rejection approach is that if there are no feasible solutions available, the algorithm may have difficulty finding the appropriate direction to move.[33]

The second type of constraint method is to use penalty functions. These can be either internal, external, or hybrid penalty functions, as illustrated in Figure 83.[212] The fundamental problem with a penalty function is that one is trying to encourage good solutions but not to the point of trapping the algorithm before it can explore other regions of the space. For this reason, the selection of $g(x)$ is often problem dependent.[232] Additionally, one must iteratively solve the system in order to find solutions at the boundary. External penalty functions iteratively increase $g(x)$, which effectively discourages infeasible solutions, but if stopped too early will result in an infeasible solutions. Internal penalty functions take the opposite approach by imposing the penalty in feasible space, but this means solutions could further improve if the algorithm was appropriately constrained. The third classic type of penalty function is to use a hybrid method which must be iterated like internal and external methods but will always have the best function value at the boundary.

Iteration is generally not a concern with metaheuristic methods such as GAs, as this can be related somehow to generation, but the selection of $g(x)$ is still problem dependent. Each of the three "classic" penalty function types described are dynamic methods, meaning the penalty changes as a function of the iteration or generation. Alternative, static methods have also been proposed which instead assign some constant penalty to infeasible solutions [157] or simply rank infeasible solutions lower than feasible ones[48], though a variety of alternative penalty function approaches have been proposed.[126] Simpler approaches have also been proposed which take $g(x)$ to be simply the sum of any constraint violations. This approach has been implemented with some success for both single- and multi-objective problems. A more complex approach, though one that has gained some significance, is to use co-evolutionary GAs. This approach in essence has two populations. One is used to solve the penalty function constrained optimization problem and the second is used to determine the best settings for penalty function coefficients.[34] While interesting and theoretically a robust approach to penalty function coefficients, this approach would likely be a more complicated than necessary approach to solving the problem.

Much like rejecting infeasible samples, a penalty function method requires the ability to evaluate infeasible solutions, meaning constraints cannot be dictated by failure regions

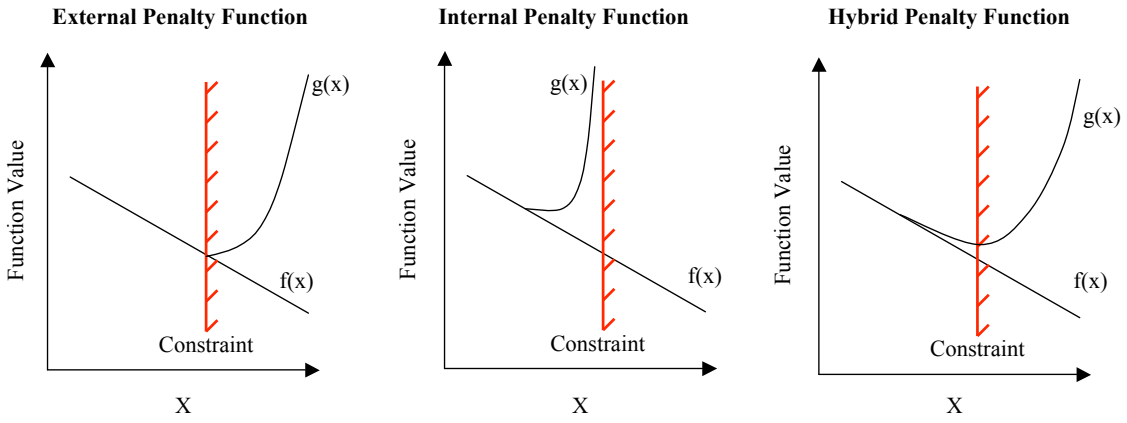


Figure 83: An Illustration of the Three Types of Penalty Functions.

for analysis codes. Unlike rejecting infeasible solutions, the penalty based approach likely encourages the algorithm to move in an improving direction even if all samples are infeasible, though this is largely dependent on how the penalty function is applied.

The third type of constraint handling found in the literature is to extend multi-objective methods. In multi-objective problems, several inputs are maximized or minimized to identify some nondominated front. These approaches add an additional objective to the optimization problem, where this additional objective is the sum of squares for all constraint violations.[10, 74] A multi-objective approach would seek to minimize this sum of squares term and would in essence allow the user to trade the constraint with other objectives. This would be particularly useful in situations where the constraint is soft, meaning it is not fixed, and would allow the user to determine how feasible and stringent the constraint is on the system at hand. This method works well for highly constrained problems because it translates the constraint into an additional objective for the problem.

The fourth type of constraint handling is to employ some sort of repair algorithm to "fix" an infeasible solution, making it a feasible solution. This repair algorithm may work well, but some have shown it is more computationally expensive to "fix" the solution than to solve the problem at hand.[33, 153] These methods were initially developed to "fix" solutions to cheap functions which can be easily repaired and have a highly constrained space.[35] These methods have been used successfully on problems such as the Traveling

Salesman Problem (TSP), the Quadratic Assignment Problem (QAP), and the set covering problem, to name a few.

Specialized operators are the fifth type of constraint handling and are commonly include methods where some problem specific information can be used to more effectively characterize the problem. For example, one may choose to impose some initial screening to remove infeasible options in a combinatorial problem. While this may reduce the space, the cost of doing this may be excessive. Specialized operators are often very helpful, but due to their problem-specific nature, they must be tailored for each application.

The last type of constraint handling is behavioral memory. Behavioral memory refers to a concept proposed by Schoenauer and Xanthakis which essentially evaluates each constraint individually, but does so in some specified order.[188] This method was shown to work well for certain test problems, but will likely "bog down" as the number of constraints increases.

After comparing these methods, the author conducted a trade study considering the two most promising methods: that of rejection sampling and the use of penalty functions. While a multi-objective approach may be of interest for soft constraints or completely infeasible problems, it was not implemented for this study, though is likely of great use for other studies.

When implementing a tournament, elitist GA, a rejection approach (or "death penalty" function) was taken which uses a static penalty function. Rather than implementing this as one would typically do in a penalty function approach ($h(x) = f(x) + g(x)$ where $g(x)$ is some penalty applied), the static penalty function was instead taken to be that in Equation 29. One should note that while this is similar to the penalty function proposed by Deb[48] and Morales & Quezada[157], it differs because this approach does not assign any value from the funtion to infeasible solutions and does not require additional coefficients to guide infeasible solutions towards feasible ones.

$$h(x) = \begin{cases} f(x), & \text{if all } g(x) \leq 0 \\ \sqrt{\sum_{\text{active } g} g(x)^2}, & \text{if any } g(x) \geq 0 \end{cases} \tag{29}$$

When performing a tournament selection, the following rules were implemented[48]:

1. If comparing two feasible solutions, the solution with the highest fitness is preferable
2. If comparing a feasible and infeasible solution, the feasible solution is preferable
3. If comparing two infeasible solutions, the solution with the lowest static penalty function value is preferable

Elitism was conducted in a similar manner. Solutions are broken into two groups; one group for feasible solutions and the second for infeasible solutions. The feasible solutions were sorted by their corresponding fitness function and the infeasible solutions were sorted by their static penalty function value, where those closest to the feasible space are preferable. When selecting the elite population, all members from the feasible group were given preference over those from the infeasible group. This tournament and elitism implementation will allow solutions to quickly move towards the feasible region and then conduct any analysis necessary. This approach will work whether any members of the initial population are feasible or not. Additionally, given a static penalty function implementation as a sum-of-squares, it is problem independent so long as the constraints, $g_i(x)$, are normalized consistently.

7.7.1.2 Ant Colony Optimization Constraint Handling

ACO constraint handling is done in much the same way that GA constraint handling is done. Because ACO is continually generating new ants, a rejection sampling variant is the best performing approach. Trade studies were conducted by the author using rejection based approaches and static and dynamic external penalty functions; the best performing method was to use a static penalty function and rejection hybrid method. Elitism was found to be very beneficial, and therefore, inferior ants were rejected each generation. Additionally, a scheme using the elite ants to update the pheromone matrix was used. The ants' objective function was evaluated using the hybrid objective function described in Subsubsection 7.7.1.1. This function was used to update the pheromone matrix as necessary using Equation 24. When selecting elite ants, an elitist strategy similar to that outlined in Subsubsection 7.7.1.1 was used. If feasible ants were available, the best selected feasible

ants were used to update the pheromone matrix. If there were no feasible ants, then the ants with the lowest hybrid function value (sum-of-squares for active constraints) were used. This was done in an effort to "guide" infeasible ants to less infeasible regions (and hopefully feasible regions).

7.7.1.3 Particle Swarm Optimization Constraint Handling

When discussing constraint handling for PSO, some differences must be drawn between approaches used for a GA or an ACO. A rejection, or "death penalty", scheme is not possible because the population members are not killed or removed; instead, they "fly" around the input space seeking to increase their fitness. Many of the constraint handling methods discussed earlier, with the exception of rejection schemes, can be implemented given some modification for PSO. These have been documented in the literature, including penalty function schemes [232, 97], repair algorithms [170], and multi-objective algorithms [10, 74], among others.

These approaches were considered and a static penalty function scheme much like that used for GA and ACO implementation was used. This approach does not take advantage of the rejection scheme used for GA or ACO, but the static penalty function, when appropriately normalized, will consistently drive toward feasible space, where an unconstrained algorithm can then be used. In order to promote convergence on the feasible region of the space, a fully connected neighborhood topology was adopted. When solving multi-objective problems, a single fully connected topology is used; however, once feasible solutions have been identified leaders will be identified and related to points in much the same way that would be used for an unconstrained problem.

7.7.2 Metaheuristic Methods Conclusions

This section has served to discuss how metaheuristic methods might be implemented to solve the RCSP. It should be noted that while only GA, ACO, and PSO will be examined any further, there are other methods such as Greedy Randomized Adaptive Search Procedure (GRASP), Tabu Search (TS), and Harmony Search (HS). These are examples of either gradient- or quasi-gradient-based methods which were not considered but do fall into the

same class of metaheuristic processes.[36]

7.8 Comparison of Constrained Discrete Optimization Methods

In order to compare the methods at hand, another qualitative comparison was performed. The criteria which were selected by the author to be significant in determining which discrete optimization method should be used were the following:

1. Nonlinear Problem Compatibility
2. Time Cost
3. Difficulty to Implement
4. Constraint Implementation

The first criteria, ability to handle nonlinear problems, is extremely important. As discussed earlier, the FSP's epistemic model uncertainty, whether treated as a side constraint or objective, is non-additive; therefore, the method must be able to adequately handle this. The second consideration, time cost, is also significant. This is meant primarily to address the number of function calls required for convergence for the method in general, not as a function of constraints or other parameters. The third criteria identified was the difficulty to implement. This is one parameter, though not necessarily a defining one, that should be considered because elegant solutions are generally simple ones, and if a simple approach and a cumbersome approach are considered, one would be foolish to choose the cumbersome one if this were the only consideration. The fourth criteria is the method's constraint handling. This includes both how well it will handle multiple constraints and how well the method can be modified to consider constraints (metaheuristic methods). As initially formulated, the FSP allows users to place constraints on a variety of different problems. As identified in various literature sources, methods with different fundamental approaches respond differently as the number of constraints increases.[17] The resulting qualitative comparison for these methods can be seen in Figure 84.

In Figure 84, both exact and metaheuristic methods for solving RCSP are compared. The first four methods are exact and the last six are metaheuristics. With respect to the first

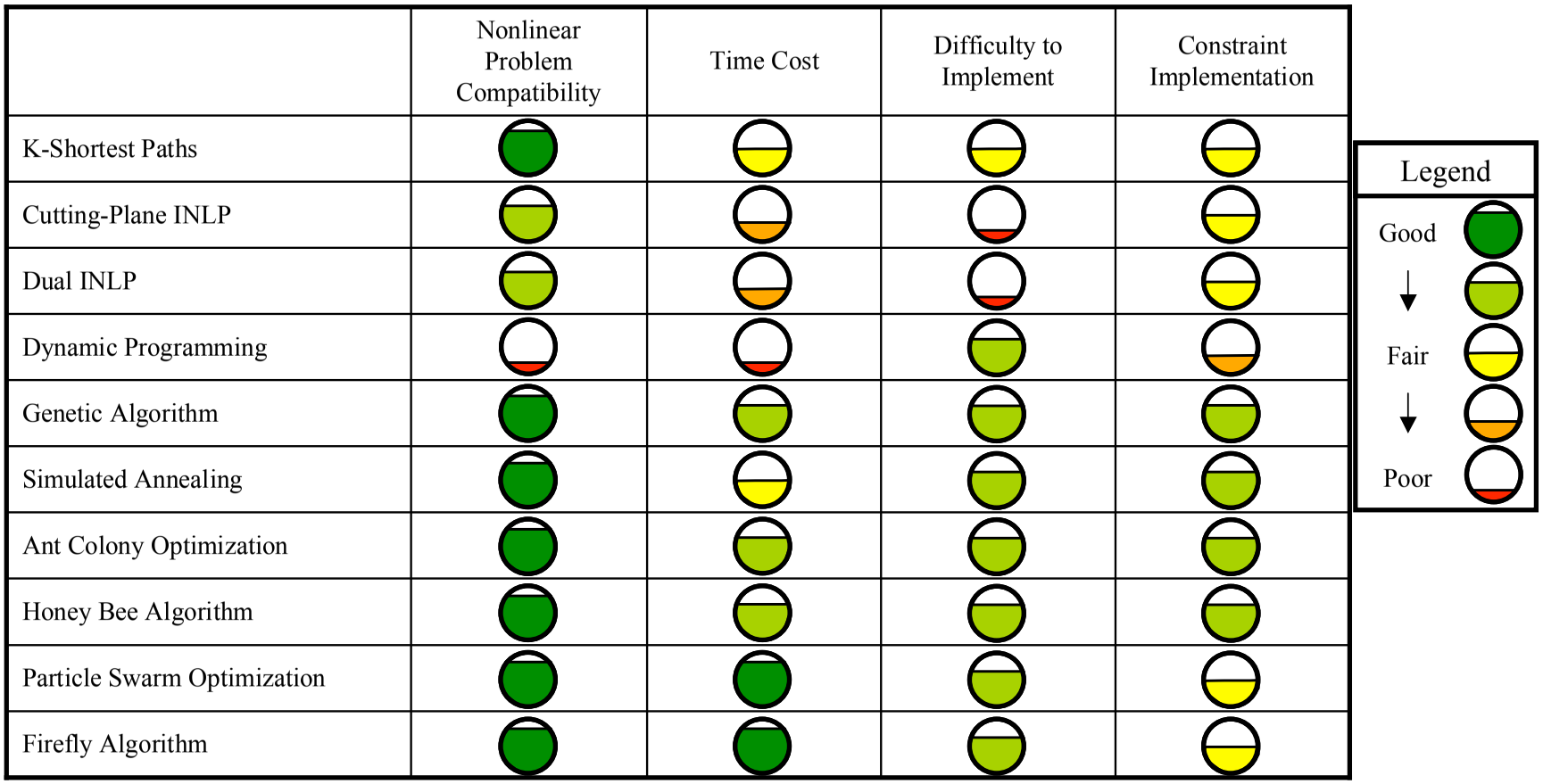


Figure 84: Qualitative Comparison of Various Discrete Optimization Methods for Solving FSP.

criteria, all methods are compatible with nonlinear problems with the exception of dynamic programming. As stated earlier, dynamic programming solves algorithms by recursively solving subproblems and then adding these optimal sub-solutions together. If these sub-solutions cannot be added, the method is actually less efficient than an exhaustive approach. Moving on to time cost, we see a strong difference between the INLP methods (both primal and dual) with respect to the other processes. Because of the exact formulation, there are a significant number of function calls required to solve the nonlinear problem using a cutting-plane approach. Comparisons between exact and metaheuristic methods have been performed for problems including the resource constrained project scheduling problem and nonlinear reliability engineering problems, which have shown a reduction in runtime when comparing ACO to plane cutting and other dual methods.[151, 192] It should also be noted that, as pointed out in the literature, there is a notable difference in convergence time (and function calls) for SA and the other metaheuristic methods.[62, 214]. Finally, K-shortest paths scored well here, but it's time cost is independent of the difficulty of the problem, as it is actually set primarily by how stringent the constraints are and the size of the problem. As stated earlier, RCSP literature states that for relatively small graphs, K-shortest paths performs moderately well because there aren't as many possible combinations and the overhead of an intelligent selection costs more than just evaluating the next shortest path.

Moving on to the third criteria, we see a strong difference between INLP and other methods. This is again due to the nonlinear and extremely problem-specific nature of INLP problems. As stated in metaheuristic literature, exact methods are extremely problem specific and difficult to solve, whereas metaheuristic approaches require comparatively little modification to solve different problems; this fact led to the rapid rise in popularity of metaheuristic methods in the 1990's (and 2000's).[134] Finally, when comparing these methods with respect to robustness with multiple constraints, K-shortest paths and dynamic programming score lower than other methods. This is largely due to how they evaluate the problem. INLP and metaheuristic methods use exact formulations or swarming properties

to solve the problem so having additional constraints will slow the algorithm, but not significantly. Dynamic programming solves RCSP by forming a Pareto frontier and then selecting the solution which best meets the requirements. As has been shown in multi-criteria literature, forming multi-dimenional Pareto frontiers becomes more difficult as the number of dimensions increases. Therefore, given how dynamic programming approaches the problem, the number of calculations significantly increases with the number of constraints. Similarly, if there are multiple or relatively stringent constraints, K-shortest paths must examine more of the shortest paths to find the optimal one.

Based on the information gained in this literature review, an experiment should be conducted which determines whether GA, ACO, and a discrete form of PSO can consistently find the optimum to the FSP.

7.9 Discrete Optimization Summary

In this chapter, relevant discrete optimization literature was examined. First, the question of how the objective function should best be formulated was addressed. This was followed by a discussion on why the FSP is a discrete, constrained problem in need of optimization. Popular discrete optimization methods including SPP and RCSP were examined and their performance for evaluating the FSP was established. Because of weaknesses in many traditional RCSP formulations, potential modifications as well as approximate metaheuristic methods were identified. Finally, a comparison of these methods based on literature was made and the three most promising methods were identified. The following chapter will discuss the experimental implementation of these three methods to solve the FSP.

CHAPTER VIII

QUANTITATIVE COMPARISON OF PROMISING DISCRETE OPTIMIZATION METHODS FOR SOLVING THE FIDELITY SELECTION PROBLEM

8.1 Introduction

This chapter will implement the three most promising optimization methods identified in Chapter 7 to solve the Fidelity Selection Problem (FSP) in experiment 4. This experiment will determine which (if any) of the methods consistently finds the correct answer to the FSP. In order to find the optimum, it is important to realize what the optimization method is doing, rather than simply treating it as a "black box". These algorithms will show why a user would pick one option for a Contributing Analysis (CA) over another. This problem can be reduced to a trade-off between the uncertainty one has in the result from an analysis and the amount of time required to get that result. In Chapter 2, the uncertainty in analyses was more specifically described and said to be epistemic model uncertainty. Based on the sparsity of data to quantify this uncertainty, an evidence theory approach was adopted to represent epistemic model uncertainty. Chapters 4 and 6 discussed possible implementation alternatives for quantifying epistemic model uncertainty and at the conclusion of Chapter 6, the most cost-effective methods were identified and the situations when the user should pick one over another were clearly stated.

This chapter will implement the most promising discrete optimization methods described in Chapters 5 and 7 to solve the FSP. These methods include a Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO). As discussed in the previous chapter, the FSP is a constrained, discrete, nonlinear, Nondeterministic Polynomial (NP)-hard optimization problem. In this chapter, a discussion of the scope of FSP will be given. This will also serve to discuss how the problem was simplified into a

manageable academic function form of FSP. This approach will allow one to more effectively demonstrate solver fitness because of the gamut of possible scenarios which can be examined. In order to assess solver fitness, 15 scenarios will be identified which are intended to encompass the best- and worst-case scenarios for FSP. The various optimization methods will undergo some tuning to determine what their "best" settings should be to solve FSP. Following this "tuning", their performance will be assessed for the FSP scenarios. This chapter will then conclude by reviewing the results and making a final suggestion on which algorithm should be used to solve the FSP.

8.2 Confirmation of Uncertainty – Time Cost Trend

Many statements have been made in this document about quantifying model uncertainty and the existence of a fundamental trade-off between model uncertainty in a result and the time cost associated with obtaining that result. However, no confirmation has been performed to ensure that this trend does exist. To illustrate model uncertainty, one could think of the Navier-Stokes equations provided in Equation 30.[7]

$$\begin{aligned}
\frac{\partial(\rho u)}{\partial t} + \nabla(\rho u V) &= -\frac{\partial p}{\partial x} + \rho f_x + (F_x)_{viscous} \\
\frac{\partial(\rho v)}{\partial t} + \nabla(\rho v V) &= -\frac{\partial p}{\partial y} + \rho f_y + (F_y)_{viscous} \\
\frac{\partial(\rho w)}{\partial t} + \nabla(\rho w V) &= -\frac{\partial p}{\partial z} + \rho f_z + (F_z)_{viscous}
\end{aligned} \tag{30}$$

These equations are a common name for the momentum equation when viscosity is considered.[7] Examining these equations, there are five basic terms. The left-most term describes momentum caused by a change in pressure. The second term describes momentum caused by changes in velocity. The third term describes changes in pressure in a given direction. The fourth term describes body forces, and the fifth term describes viscous forces. While solving the complete Navier-Stokes equations would be quite difficult, one can assume that certain terms (for example the body forces and viscous forces terms) are unimportant. This reduced form can be evaluated. While the result is not as accurate as it would be if these terms were included, the time cost is less. This is the basic argument that has been made, but no computational data has been provided to support it.

Next, one can to take this same concept of adding or removing terms to trade off computational cost with the quality of an answer produced and apply it to an engineering example. The difference for this illustration is that rather than adding or removing viscosity or momentum terms, the model coefficient will be removed. For this confirmation, the computational cost with evaluating a Taylor series expansion will be traded with the ability of the resulting expression's ability to predict data from an engineering analysis. The engineering analysis (truth data) for this illustration was taken to be powerhook data for a turbojet engine operation at Sea-Level Static (SLS) conditions.

A powerhook is a plot of net thrust vs. Thrust Specific Fuel Consumption (TSFC) where engine performance is provided for all conditions from maximum to minimum power setting. The powerhook considered for this confirmation was produced by running an engine simulation; however, it can be approximated with a Taylor series expansion. As the number of terms in the Taylor series expansion increases, the resulting equation is able to better approximate the powerhook data, but this comes at some cost as more operations are required to evaluate the expression. This trade-off is provided in Figure 85 and the resulting representation, with its uncertainty, is provided in Figure 86.

The results provided in Figure 85 clearly show the anticipated trend: to get a more accurate answer, one must spend more time performing the computation (including more terms). The order for approximation is the largest exponent on the independent variable; for this example, the independent variable is net thrust. Therefore, a 0^{th}-order approximation is simply a constant. A 1^{st}-order approximation is linear, while a 2^{nd}-order approximation is quadratic, and so on. The number of operations was calculated using the following rules: evaluating a constant is a single operation, and every addition, subtraction, multiplication, or division is an additional operation. Figure 85 serves to confirm the expected trade-off between model uncertainty and time cost for obtaining a result.

The results in Figure 86 illustrate how the uncertainty in each evaluation changes as number of terms in the approximation (and equivalently the number of operations) increases. In this figure, the actual powerhook data is provided with blue samples, the model prediction is shown with a black line, and the uncertainty in this prediction is shown with a grey band.

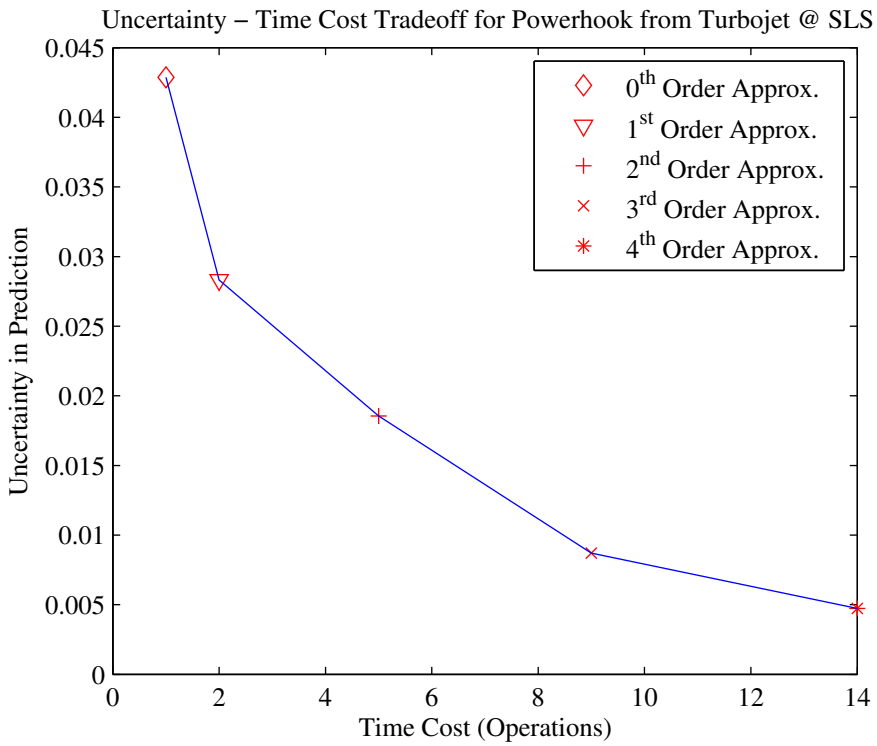


Figure 85: Trade-off Between Uncertainty and Operations for Representation of a Turbojet Operating at SLS

8.3 Fidelity Selection Problem Scope

When considering the FSP, one is fundamentally determining why one would select a given tool option over another on some Pareto efficient frontier of tool choices. This is illustrated in Figure 87. In this figure, a notional gas turbine engine multi-fidelity Design Structure Matrix (DSM) is provided. Examining a given Contributing Analysis (CA) in the DSM, one can see a variety of choices. Before integration, one has a priori knowledge about the cost of running the given analysis, its time cost, and the amount of uncertainty it has in some output from the code (not necessarily the output of interest). While this is useful information, it is only part of the information necessary to solve the FSP. For example, to trade-off between the green and blue analysis options in Figure 87, one must know not only the time cost and uncertainty each has in a compressor output (known a priori), but more importantly the aggregate run time for the analysis given any feedback loops which may occur and the uncertainty this option propagates to some gas turbine output of interest.

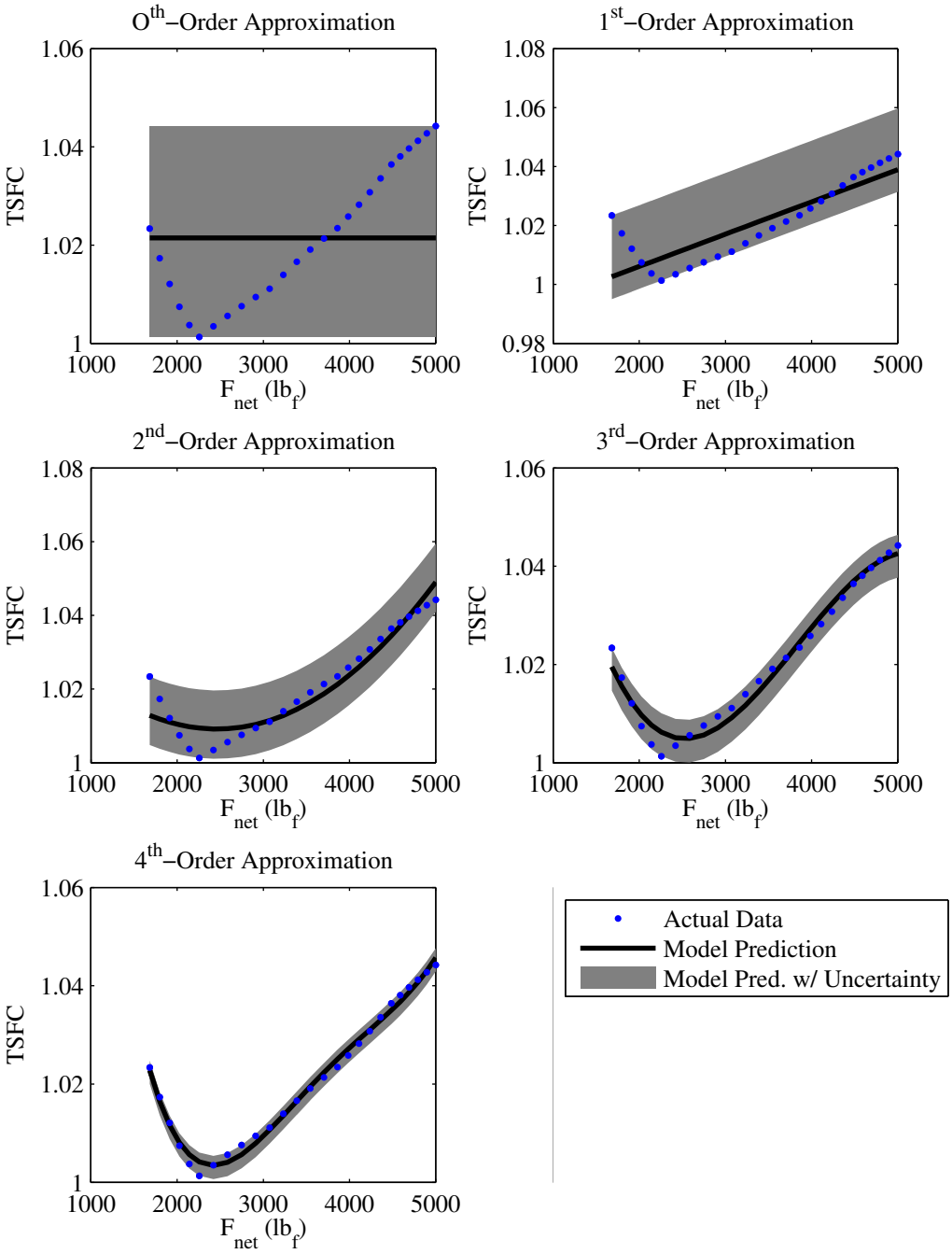


Figure 86: Uncertainty in Various Taylor Series Approximations for a Turbojet Powerhook at SLS.

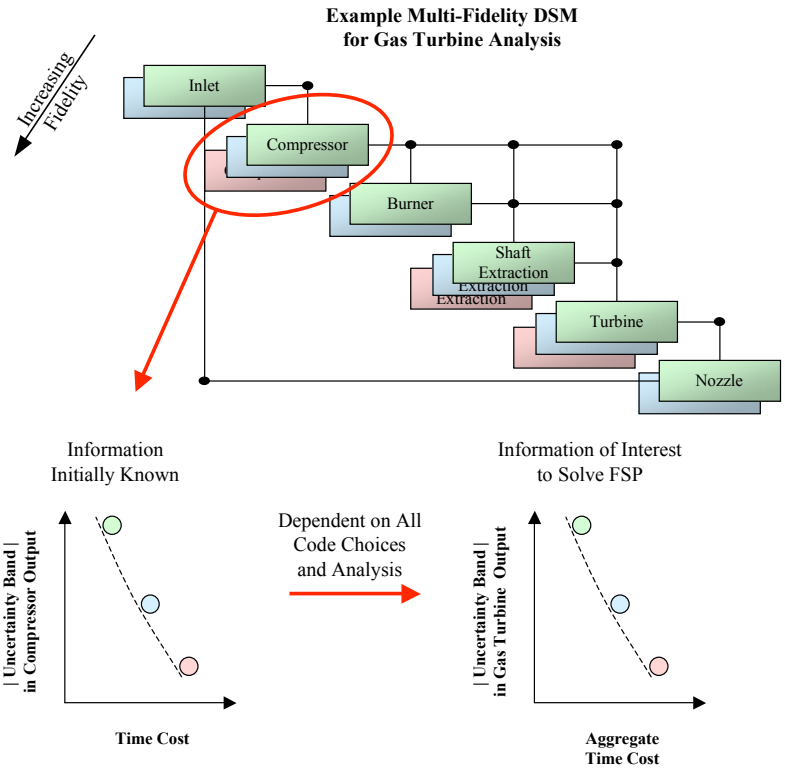


Figure 87: Notional Illustration of FSP.

Unfortunately this more useful information is not known and is also dependent on other analyses which must be chosen. The information which is known initially is shown in the lower left corner of Figure 87 and the information which is actually needed to perform the selection is provided in the lower right corner of the same figure.

Obtaining this uncertainty and the aggregate run time are dependent upon two things not initially known when the tool selection is made:

1. The uncertainty of any inputs it is being provided

2. How the uncertainty from the choice will be transformed to the output of interest

If these two points were not clouding the tool choice, then it could be much more easily conducted. Instead, one must select all choices and actually conduct an analysis to achieve some result. Additionally, it means that one should take caution when choosing a problem to select the best optimization method for solving the FSP. If the problem were inappropriately chosen, one would simply be making a statement about how the FSP can be

solved for a class of systems which has that specific uncertainty – time cost trade-off series of choices and Pareto frontier shape. Additionally, if the solver was actually underperforming in a given dimension which happened to correspond to some comparatively insignificant CA in the analysis, this would not be known. Therefore, it would be extremely desirable if one could decouple the mechanism for choosing an FSP solver from the problem being addressed.

Decoupling the mechanism for selecting the FSP solver from the problem being addressed is the approach that was taken for this research. This results in developing a canonical form for the FSP, which is a framework that allows one to vary all settings corresponding to the characteristics of the uncertainty – time cost Pareto frontier. After doing this, a brainstorming exercise was conducted to identify possible input variable types for this decomposed FSP. The results were as follows:

1. Number of CAs in the DSM
2. Number of Choices for Each CA
3. Shape of the CA Pareto Front for Each CA
4. Spacing of the Choices on the Pareto Front for Each CA
5. Time Cost Scaling for Each CA
6. Uncertainty Scaling for Each CA
7. Location of Constraint

The first input variable type, the number of CAs, is the number of uncertainty – time cost curves that exist in the DSM. This is significant because it represents the number of input variables for the uncertainty quantification and fidelity selection. As illustrated in the uncertainty quantification portion of this document, the number of input variables significantly impacts which uncertainty quantification method should be chosen and how long it will take to reach an answer. The uncertainty – time cost curve will change as the number of CAs increases as illustrated in Figure 88.

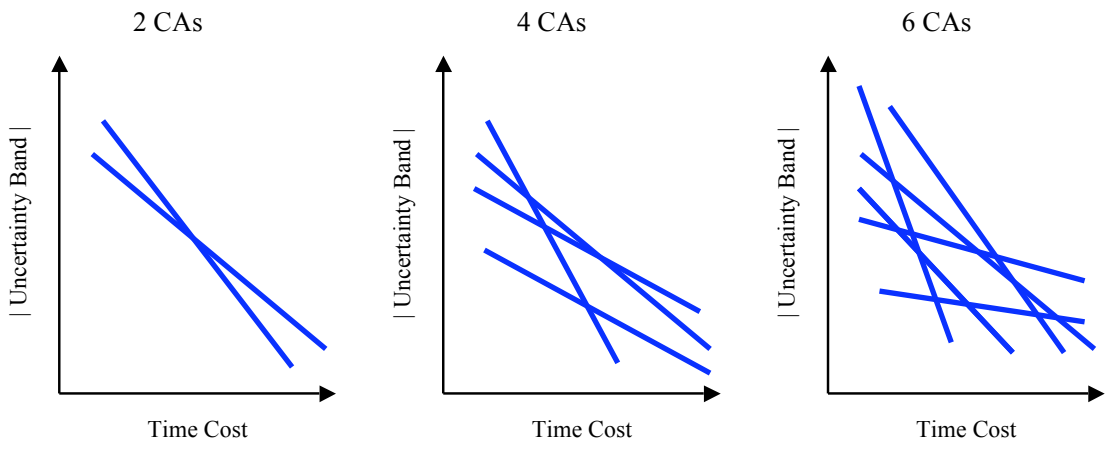


Figure 88: Illustration of Uncertainty – Time Cost Trade-off for Various Numbers of CAs.

The second input variable type for the FSP is the number of choices for each CA in the DSM. This variable type must be defined for each CA in the DSM. This is significant because it determines how many settings there are on each dimension. For a GA this determines the binary string length. For an ACO implementation, this determines the number of choices the virtual ants can make, and for PSO, this determines how many different settings the swarm population members will be rounded to. This is graphically illustrated in Figure 89. This variable was thought to be significant because it, combined with the number of CAs, determines the size of the space. If one of the considered methods converges on a solution without exploring enough of the space or if it spreads out too much, it may not be able to find the correct solution. This trend may be seen for an increasing number of choices.

The third variable type considered for FSP is the shape of the front. This variable type must be defined for each CA in the DSM being considered. As choices are spaced along the front, the shape of the front will drive whether moving one choice to the left or right will significantly change the result. This is significant because it is thought that it will require a more fit solver to explore a space if no change is initially seen for a given variable but there is later a large change in uncertainty or time. The various CA uncertainty – time cost Pareto frontiers are illustrated in Figure 90.

The fourth variable type examined is the spacing of the CA choices along the uncertainty

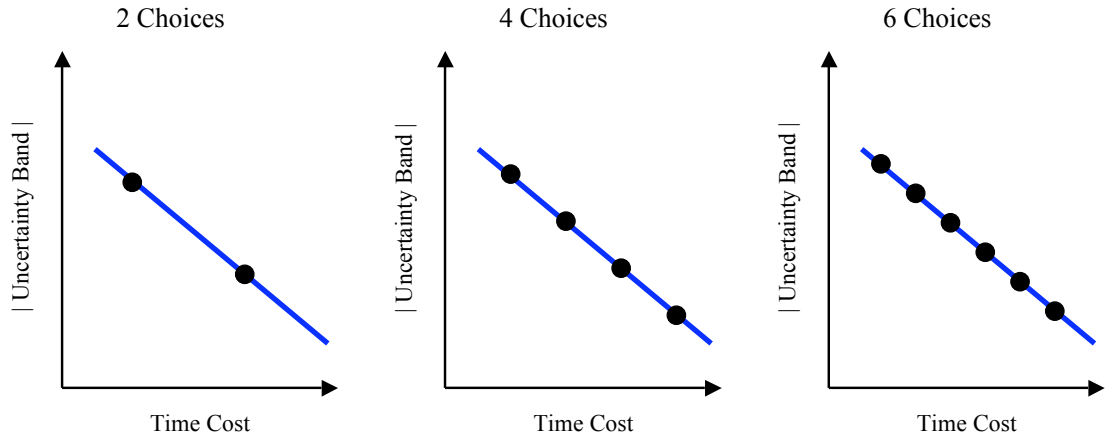


Figure 89: Illustration of Uncertainty – Time Cost Trade-off for Various Numbers of Choices per CA.

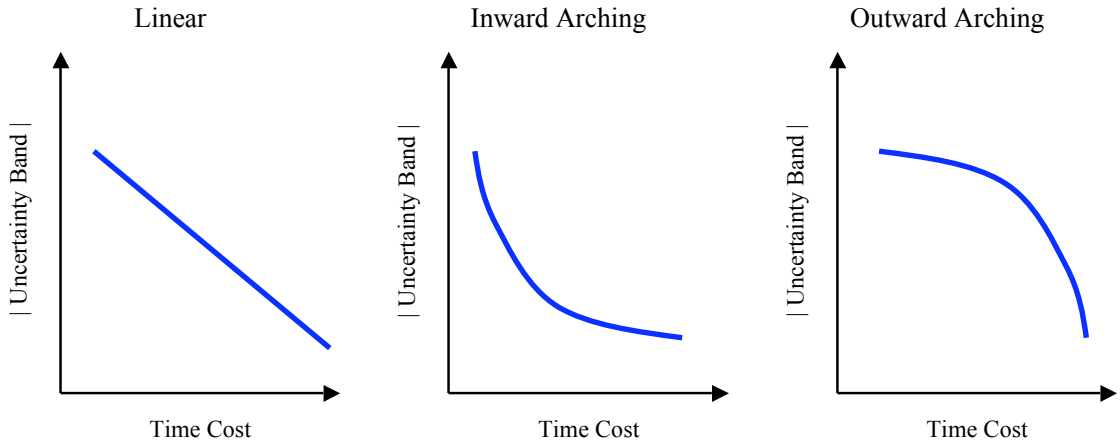


Figure 90: Illustration of Uncertainty – Time Cost Trade-off for Various CA Front Shapes.

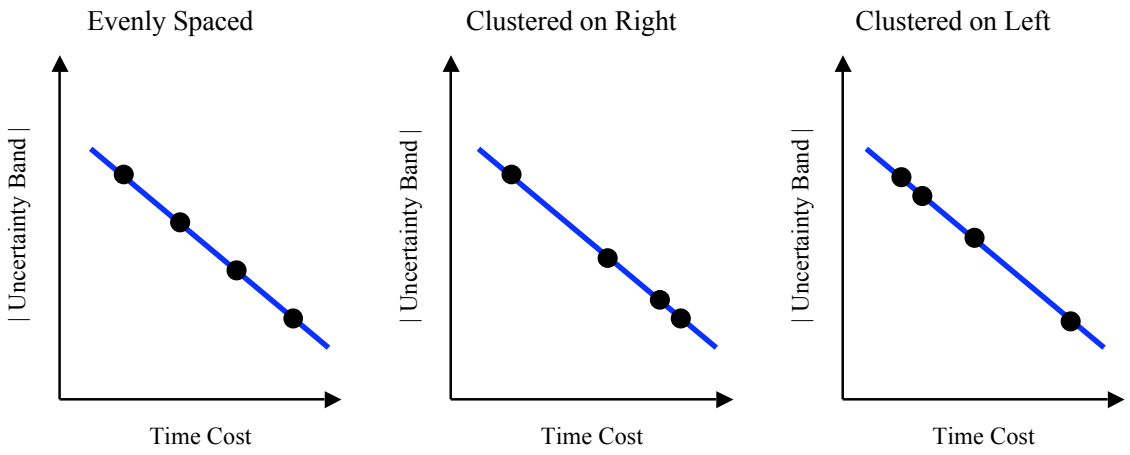


Figure 91: Illustration of Uncertainty – Time Cost Trade-off for Various Choice Spacing Scenarios.

– time cost front. This variable must be defined for each CA. It was thought that if the options were evenly spaced it would be easiest to solve, whereas if they were clustered at one end or the other, especially given some nonlinear front shape, this would make the FSP more difficult to solve. For this reason, scenarios were considered with choices clustered both at the right and left end of the front. This is graphically illustrated in Figure 91. One should note that the choice clustering is implemented independently of the front shape. In order to place choices along the front, the front is in essence "stretched out" into a linear curve, the choices are placed upon it, and then it is "bent back" into its original form.

The next two variable types considered were time and uncertainty scaling. As with nearly all of the other variable types, these must be defined for each CA in the DSM. These was added to represent how one analysis could be more "important" in calculating the uncertainty than another. By creating a scaling on the time and uncertainty axes, one can use this to create the canonical form for solving the FSP. With this canonical form, the magnitude of the uncertainty band is that of the output of interest from the simulation (shown in the lower right part of Figure 87). This is not the output variable from the CA, which will then pass into another CA in the DSM (shown in the lower left part of Figure 87). The axes scaling for time cost and uncertainty are graphically illustrated in Figures 92 and 93, respectively.

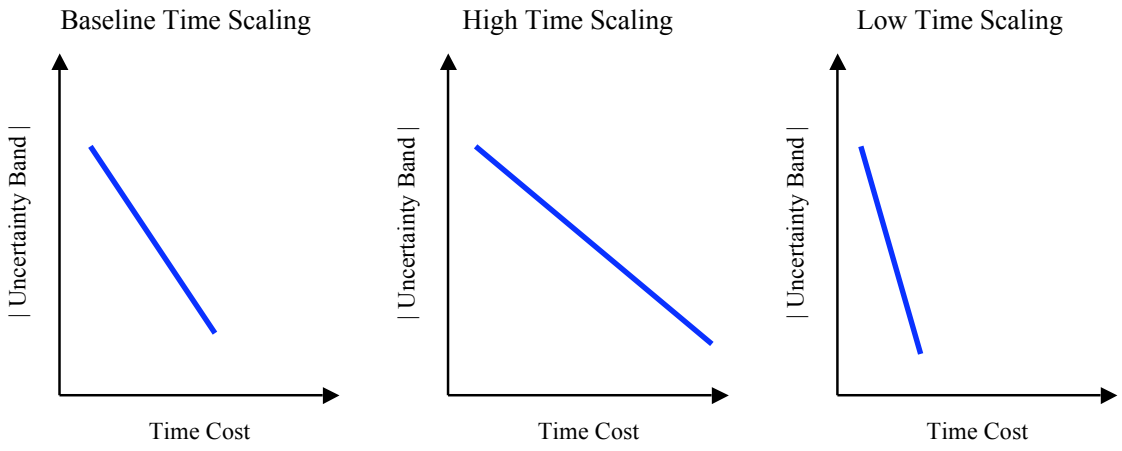


Figure 92: Illustration of Uncertainty – Time Cost Trade-off for Various Time Cost Scalings.

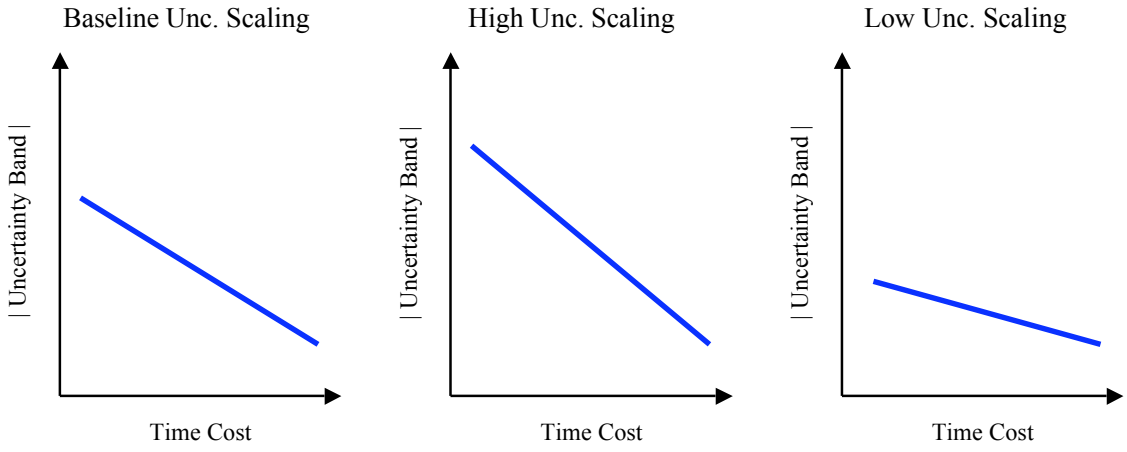


Figure 93: Illustration of Uncertainty – Time Cost Trade-off for Various Uncertainty Scalings.

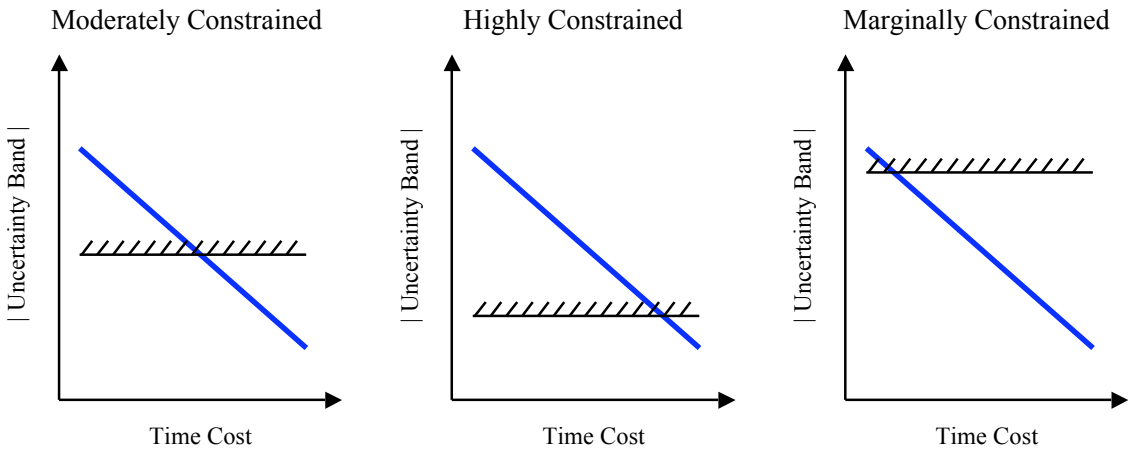


Figure 94: Illustration of Uncertainty – Time Cost Trade-off for Various Uncertainty Constraint Settings.

The seventh and final variable type considered for the FSP was that of the uncertainty constraint location. This is considered to be an important variable because it will drive the solver to a given part of the uncertainty – time cost front, which may be significant if it is nonlinear (illustrated in Figure 90). This is also significant from a solver perspective. As discussed earlier in Chapter 5, certain constrained optimization methods do not work well if the problem is highly constrained, meaning the entire initial population is infeasible. For this reason, this constraint was thought to be particularly important. This input variable's effect is graphically illustrated in Figure 94.

One final part of the FSP which has not been mentioned is the analysis being conducted. In order to assess solver fitness and decouple the FSP from the analysis being conducted, the simplest analysis possible was chosen: addition. The objective function being evaluated will use the functions provided in Equations 31 and 32.

$$U_{Total} = \sum_{i=1}^{\#CA} U_i \tag{31}$$

$$T_{Total} = \sum_{i=1}^{\#CA} T_i \tag{32}$$

Where:

- U is the magnitude of the uncertainty band in the result of interest

- T is the aggregate time cost for the CA or DSM of interest

While this is a simple function (the simplest that the author could think of), its simplicity should therefore more effectively illustrate the fitness of the FSP solver, given various possible characteristics of the multi-fidelity Modeling and Simulation (M&S) environment. Given the seven input variable types for FSP discussed here, one has a canonical form of sorts where any possible analysis and number of choices for the CAs can be mapped to a combination of these seven canonical FSP variable types. The only problem is that the user cannot actually conduct this mapping until the selection has been made. This means that in order to choose a solver, one should choose one most capable of solving *any* possible combination of these seven variables.

8.4 Reducing the Fidelity Selection Problem to a Manageable Surrogate: Scenario-Based Simulation

A canonical form for the FSP was outlined in the previous section and illustrated in Figures 88 through 94. The simple academic equation being evaluated is also provided in the previous section in Equations 31 and 32. However, even given the abstracted nature of the canonical FSP, the problem space quickly becomes unmanageably large as the number of CAs increases. As illustrated in Table 7, the seven types of variables cause the number of variables in the canonical FSP to grow as $5(\#\text{ CA}) + 1$. Data is provided in this table for a 2-level full factorial, though if one were looking to identify effects, a 3- or 4-level full factorial would be of more interest. A full factorial grows as the number of levels raised to the power of the number of variables $((\#\text{ levels})^{(\#\text{ variables})})$. Given this, the 3- and 4-level full factorials would explode even more quickly than the one provided in Table 7. Additionally, because one would want to show solver fitness with respect to increasing problem size, it is extremely important that combinations with 10 and 12 CAs be evaluated. Given the unmanageable size of this space, some alternative approach must be adopted.

One approach suggested by some for sampling highly dimensional problems when a full factorial cannot be evaluated is to use Monte Carlo (MC) techniques.[22] However, as discussed earlier in Chapter 4, this is a very poor method for accounting for any possible

Table 7: Number of Variables for Canonical FSP for Various Number of CAs.

# CAs	# Input Variables	# Samples in 2-Level Full Factorial
1	6	64
2	11	2048
4	21	$2.1 * 10^6$ (2.1 Million)
6	31	$2.15 * 10^9$ (2.15 Billion)
8	41	$2.2 * 10^{12}$ (2.2 Trillion)
10	51	$2.25 * 10^{15}$ (2.25 Quadrillion)
12	61	$2.31 * 10^{18}$ (2.31 Quintillion)

scenario and actually only serves to sample the middle of a given space very thoroughly. For this reason, some alternative approach was needed, and a scenario-based approach was adopted.

In order to adequately assess the true variability of the FSP, a series of 15 scenarios was identified as potential best- and worst-cases. These 15 scenarios were created by selecting 5 different uncertainty – time cost frontier/code spacing scenarios. These five scenarios are illustrated in Figure 95. The choices are evenly spaced along the frontier, which is linear. Moving on to the center column of scenarios, the choices are clustered at a low time-cost and high uncertainty setting. The right column contains two scenarios where the choices are clustered at a high time cost and lower uncertainty setting. The top row (center and right columns) represents scenarios where the choice front is inward arching, whereas the bottom row (center and right columns) represents scenarios where the choice front is outward arching. Depending on the constraint location (marginally, moderately, or highly constrained), some of these scenario options would be more difficult to evaluate than others.

Once these five front shape/choice spacing options were selected, there were five remaining variable types: the number of CAs, number of code choices for each CA, the axes scaling (time and uncertainty), and constraint location. The number of CAs was allowed to vary to show how the various solvers will perform for a growing problem. Each of the five front shape/choice spacing options were evaluated with 2, 5, and 8 choices per CA, making

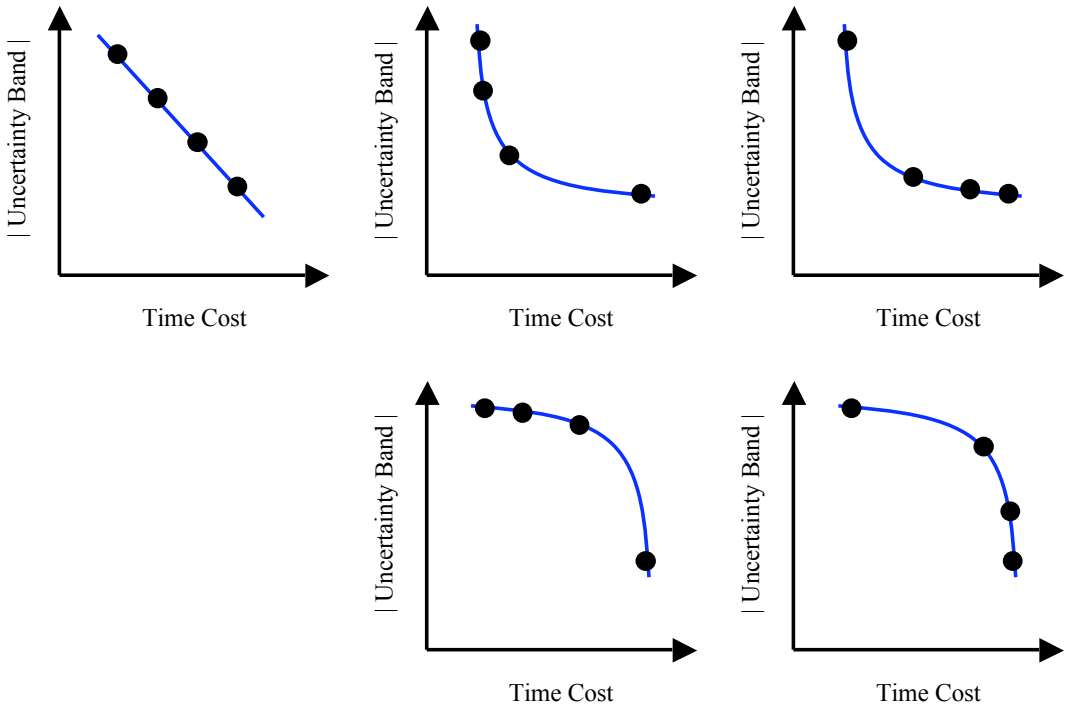


Figure 95: Illustration of Five Uncertainty – Time Cost Scenarios.

15 scenarios. In order to assess solver fitness, each dimension should be just as difficult to find the optimum in as another, and therefore no axes scaling was employed. Additionally, this means that each CA should have the same number of choices, front shape, and choice spacing. This would allow one to emphasize the ease or difficulty of evaluating a given scenario. The input variable type settings for each of the 15 scenarios are provided in Table 8.

The final input type, constraint location, was allowed to vary between three different settings: where the constraint is at 90% of the maximum uncertainty (marginally constrained), 50% of the maximum uncertainty (moderately constrained), and 10% of the maximum uncertainty (highly constrained). These three different constraint levels were investigated for each of the 15 scenarios outlined in Table 8.

8.5 Algorithmic Implementation for Promising Optimization Methods

As mentioned earlier in this chapter, a GA, ACO, and PSO were implemented to solve this constrained discrete optimization problem. These three methods were discussed in detail in

Table 8: FSP Variable Type Settings for each CA for the 15 Considered Scenarios.

Scenario	# Choices per CA	CA Pareto Front Shape	Choice Spacing
1	2	Linear	Evenly
2	2	Inward Arching	Clustered on Right
3	2	Outward Arching	Clustered on Right
4	2	Inward Arching	Clustered on Left
5	2	Outward Arching	Clustered on Left
6	5	Linear	Evenly
7	5	Inward Arching	Clustered on Right
8	5	Outward Arching	Clustered on Right
9	5	Inward Arching	Clustered on Left
10	5	Outward Arching	Clustered on Left
11	8	Linear	Evenly
12	8	Inward Arching	Clustered on Right
13	8	Outward Arching	Clustered on Right
14	8	Inward Arching	Clustered on Left
15	8	Outward Arching	Clustered on Left

Chapter 5. Chapter 5 also included a discussion on both continuous and discrete implementation for constrained and unconstrained problems. For clarity, the specific implementation being used to solve the FSP will be reviewed here.

8.5.1 Constraint and Objective Function Implementation

Based on the findings discussed in Chapter 5, the constraint will be enforced through both a rejection and penalty function approach. The GA and ACO will use elitism, meaning half of the population is eliminated each generation and the other half is allowed to move on to the next generation. This approach will improve convergence because unlike traditional implementations, the best solutions will move on to the following generation and will only be replaced if better solutions are found. The GA will use a tournament selection each generation, and this is also a rejection approach. When implementing the penalty function, which will be used by all three methods, the functions provided in Equations 33 and 34 will be used.

$$h(x) = \begin{cases} f(x), & \text{if all } g_i(x) \leq 0 \\ \sqrt{\sum_{\text{active } g_i} g_i(x)^2}, & \text{if any } g_i(x) \geq 0 \end{cases} \tag{33}$$

$$g_i(x) = \left(\frac{\text{Uncertainty in } Y_i - \text{Uncertainty Constraint for } Y_i}{\text{Uncertainty Constraint for } Y_i} - 1 \right) * 100 \tag{34}$$

Where:

- $h(x)$ is the new objective function
- $f(x)$ is the sum of aggregate run time for a given tool combination
- $g(x)$ is the percent violation of an uncertainty constraint for output i

This objective function, $h(x)$, has two components. If a solution is feasible, meaning it meets all uncertainty constraints on the system, then $h(x)$ is equal to the aggregate run time for that combination ($f(x)$). If, however, the solution is infeasible, then $h(x)$ is equal to the square root of the sum of squares of the percent violation of all violated uncertainty constraints. Calculating the percent violation of each constraint serves to normalize it, making

all constraints equally as important. This does so without requiring additional weighting coefficients, as is typically done in the literature. What separates this implementation from others is that the constraints for this problem are on the output space and are independent of the location in the design space, allowing for a more straight-forward implementation of the penalty function.

When comparing solutions, preference is given to a feasible solution over an infeasible one. A given candidate's fitness relative to other feasible (or infeasible) solutions is determined by its fitness in $h(x)$. This objective function value will be used in assigning the leader in PSO, for determining which ants update the pheromone matrix in ACO, for determining which solution is most fit in the GA, and for determining the elite portion of the population in the GA and ACO.

One additional modification was placed in each of these methods with respect to the function evaluations. When these algorithms were used earlier to find the boundaries of continuous problems, it was assumed that a function evaluation was cheap. However, that is not the case for this analysis. Here, a function evaluation is expensive, and therefore data should not be duplicated. Each method was modified to store the solutions from previous runs. Therefore, a candidate tool combination would only cost a function evaluation if it had not already been considered. If it had been, then the previous result would be used.

8.5.2 Algorithmic Tuning for Simple Academic Function Fidelity Selection Problem

As was discussed in Chapter 5, when one chooses to use a metaheuristic method, they are also assuming that a function evaluation is relatively cheap. Typically, a Response Surface Equation (RSE) of some sort is created for the analysis. Because this is not the case for the FSP, meaning the analysis is not cheap and that an RSE cannot be created for the model, the various metaheuristic optimization methods should be re-examined to more appropriately select the population size and various other tuning parameters to more effectively find the optimum.

For this reason, a trade study was conducted. Here, the three most important parameters in each optimization method (GA, ACO, and PSO) were identified and then varied

Table 9: Full Factorial Ranges for Parameter Tuning for the Simple Academic Function FSP.

Method	Parameter	Low Setting	Middle Setting	High Setting
GA	Population	25	50	75
	P_{cross}	50%	70%	90%
	P_{mutate}	50%	70%	90%
ACO	Population	25	50	75
	ρ	0.5	0.7	0.9
	% to Update	5%	15%	25%
PSO	Population	25	50	75
	V_{cap}	75%	150%	225%
	ω	0.7	0.9	1.1

in a 3-level full factorial. As was discussed earlier, these metaheuristic methods do not consistently produce the same result. In order to gain some measure of the variability in their result, each of the 27 cases was run 10 times. In addition to repeating each point 10 times, each parameter setting combination was evaluated for various constraint settings, with the constraint moving from 90% of the maximum uncertainty to 10% of the maximum uncertainty while being evaluated at 20% increments.

The three parameters for the GA that were identified to cause the most variation were the population and the crossover and mutation rates. The ACO's greatest variability was found from the population size, the pheromone evaporation coefficient (ρ), and the percent of the population to be used to update the pheromone matrix. The implemented PSO was found to have significant variation from its population size, the maximum velocity (V_{cap}) which serves as a move limit on each point, and the momentum coefficient (ω). The ranges and settings examined for each of these parameters can be found in Table 9.

In order to tune the model, a single candidate scenario, scenario 15 with 10 CAs, was selected and each of the parameter combinations were evaluated for each model. The results from this tuning are interpreted two ways. First, the distance from the optimum was

determined and each solutions' distance from this optimum was calculated. Additionally, the number of function evaluations required was determined. The distance from optimum is provided in Figure 96. In this figure, the top left pane is the most constrained problem (the constraint is at 10% of the maximum uncertainty). The constraint progressively becomes less constraining as one moves to the right and downward. In each pane, the horizontal axis corresponds to the full factorial case number and the vertical axis corresponds to the distance from the optimum. Because each full factorial case for each constraint setting was run 10 times, the data is interpreted as the $\mu \pm \sigma$ calculated from the samples. Therefore, a good combination has a very short band that is at a distance of 0 (meaning it is at the optimum). The various colors correspond to the three methods considered.

From Figure 96, one can see that as the constraint becomes more stringent, ACO is unable to find the optimum, regardless of the parameter setting. Additionally, we can see that V_{cap} is an extremely important parameter when considering PSO convergence. Looking at the PSO convergence, the first nine cases correspond to the low setting for V_{cap}, the next nine correspond to the middle setting, and the last three correspond to the highest setting. When at the low V_{cap} setting, PSO is not necessarily able to reach the optimum. However, as one looks at the middle and high settings, PSO is clearly able to find the optimum. One interesting note is that the GA is able to consistently converge to the right answer regardless of the parameter or constraint setting.

While the distance from the optimum is very significant, one should remember the purpose for this study is to not only increase the probability of finding the right answer, but to do so in as little time as possible. Function evaluations are no longer relatively cheap, as discussed in earlier chapters, and therefore, reducing the number of function evaluations will significantly reduce the time required to calculate the best tool combination. The results for this parameter tuning study when considering the number of function evaluations are provided in Figure 97.

The data in Figure 97 should be interpreted much like the distance data in Figure 96. The only difference is that in Figure 97, the vertical axis in each pane plots the number of function evaluations, not the distance from the optimum. Like the previous figure, this one

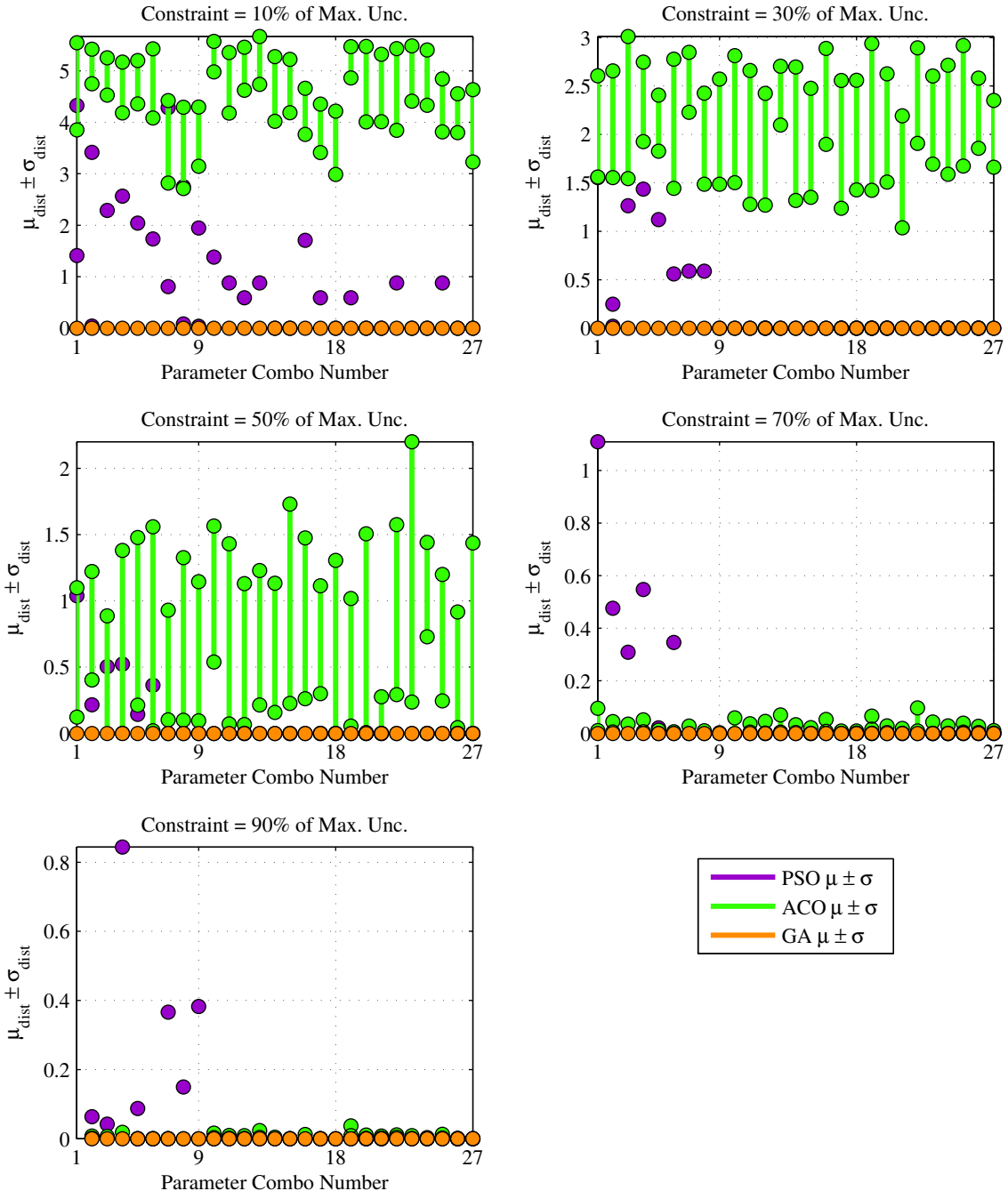


Figure 96: Distance from Optimum in Parameter Tuning Study for Various Constraints.

Table 10: "Best" Settings for Simple Academic Function FSP Optimization Alternatives.

Method	Parameter	"Best" Setting	Full Factorial Case
GA	Population	25	1
	P_{cross}	50%	
	P_{mutate}	50%	
ACO	Population	25	16
	ρ	0.7	
	% to Update	25%	
PSO	Population	50	20
	V_{cap}	225%	
	ω	0.7	

looks at the $\mu \pm \sigma$ for the 10 samples taken. From this data, one can see that while the GA is able to always reach the right answer (data in Figure 96, it does so at a higher expense than PSO. The number of PSO function evaluations is always less than either the GA or ACO, and this actually *decreases* as the constraint becomes more stringent.

From the data in Figures 96 and 97, as well as from an Analysis of Variance (ANOVA) which was conducted with this data, the "best" settings for these models were established. This information is provided in Table 10. It was initially assumed that scenario 15 was representative of the characteristics of all scenarios for the simple academic function FSP, and therefore these settings were used for all scenarios examined.

8.6 Fidelity Selection Problem Results

The goal for this chapter is to identify which method, with which what settings, can best solve the FSP or some surrogate of it. This section will detail results from an experiment created to answer this question. The canonical FSP using the simple academic function analysis will be evaluated for the 15 scenarios identified in Section 8.4. These 15 scenarios will be evaluated at three different constraint settings: one where the constraint is at 90%, a second at 50%, and a third at 10% of the maximum uncertainty possible for the given

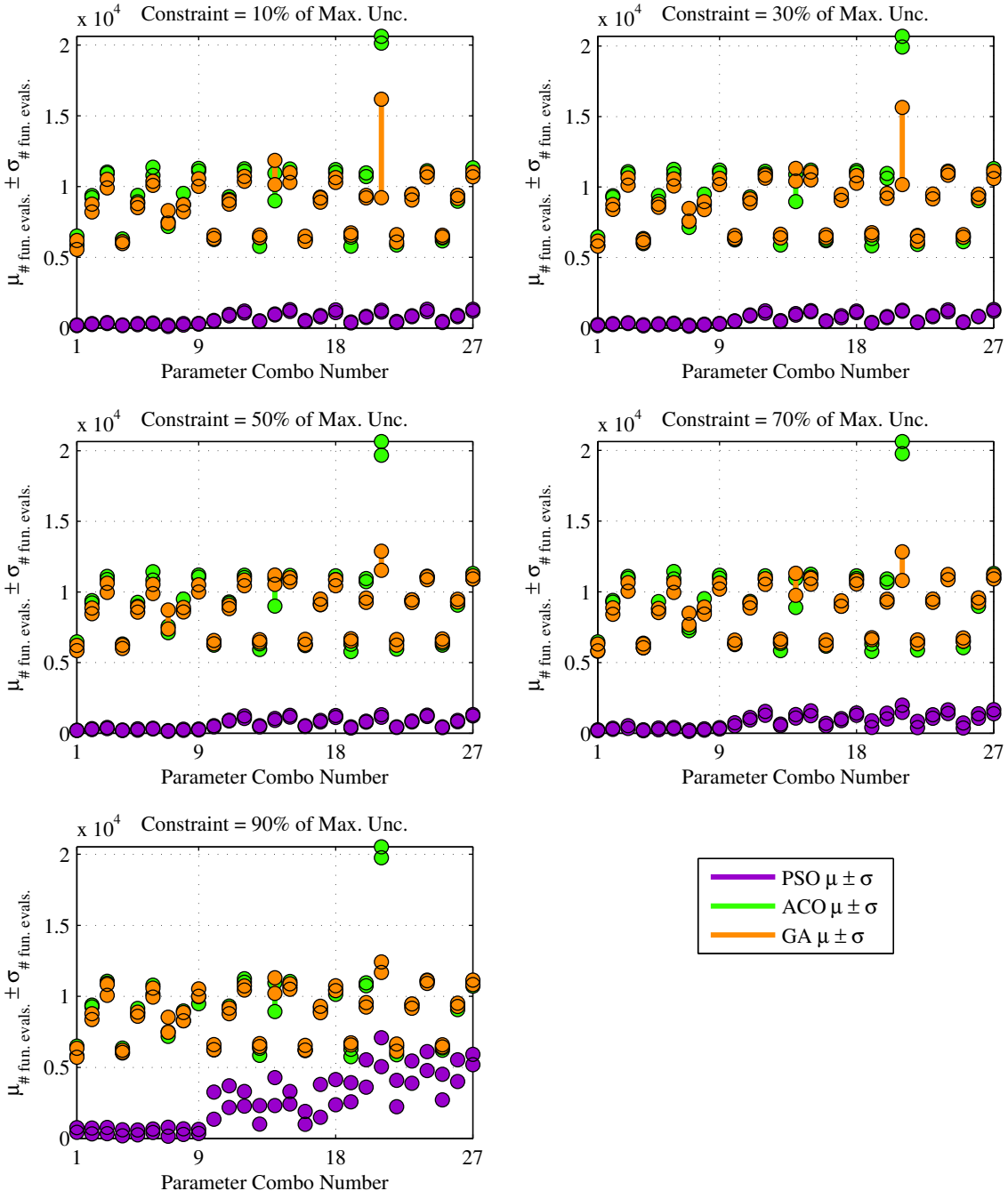


Figure 97: Number of Function Evaluations in Parameter Tuning Study for Various Constraints.

DSM. Because many methods have a tendency to perform better for lower dimensional problems than higher dimensional ones, each scenario – constraint setting combination will be evaluated with 4 to 12 CAs. Additionally, each of these methods are stochastic by nature, and one does not necessarily have a lot of confidence that the optimum has been found in a given evaluation. For this reason, each CA/scenario/constraint setting combination will be evaluated 10 times to gain some measure of its variability (and consistency in finding the optimum).

In the parameter tuning study conducted, the goal was to identify a method with a low variability which consistently reached (or nearly reached) the optimum for the system. This led to the selection of the "best" settings for the GA, ACO, and the PSO. When interpreting these methods' performance, a slightly different approach was taken. Here, the results are interpreted by examining the $\mu \pm \sigma$ for the distance from the "optimum" and the number of function evaluations, similar to what was done earlier, and also whether or not the method found the "optimum" for the study.

This "optimum" differs from a true optimum. The true optimum can only be found by evaluating all possible choices for a given DSM. As stated earlier, the problem was examined when with 4 to 12 CAs. For some of the scenarios, specifically scenarios 11 through 15 which have 8 choices for each CA, for large problems (10 to 12 CAs), all possible combinations cannot be calculated. For this reason, the following alternative approach was taken. For a given scenario, constraint setting, and number of CAs, the best result obtained by any of the three methods for any of their 10 evaluations was treated as the "optimum". While this may not be the actual optimum for the problem, it could be thought of as a relative optimum, and at the very least, the point closest to the optimum which was obtained.

First, interpreting whether or not a given method achieved the optimum, the reader should consider Figure 98. Here, various histograms are provided. Each pane is for a given number of CAs. The vertical axis (or height of the bar) is the percent of scenarios evaluated, for that number of CAs, where the various methods found the "optimum". The horizontal axis in each pane is for the constraint setting. The left most grouping in each pane is for the highly constrained condition, where the constraint is at 10% of the maximum

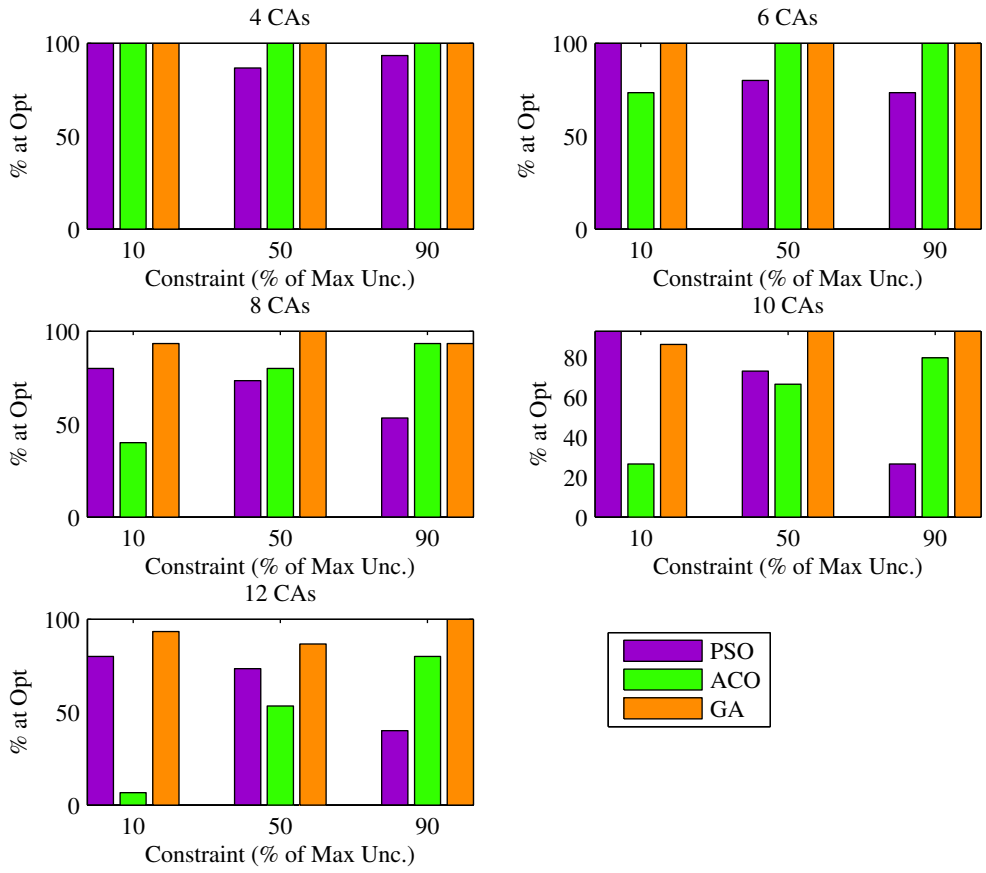


Figure 98: Academic Function FSP Results – Percent of Iterations Where Methods Found the "Optimum", Sorted by Constraint and Number of CAs.

uncertainty. The middle is at 50% of the maximum uncertainty, and the right most grouping is for the marginally constrained condition, where the constraint is at 90% of the maximum uncertainty. In each of these panes, PSO is shown in purple, ACO is shown in green, and the GA is shown in orange.

Interpreting these results, one can see that PSO found the correct answer most often for the highly constrained instances, and as the problem became less constraining or increased in dimensionality, PSO was less likely to find the correct answer. Moving on to ACO, we see the opposite occurring. ACO performs very well in marginally constrained instances and becomes less likely to find the right answer as the problem becomes more constraining or as the dimensionality of the problem increases. One should note that while this is the opposite trend than that seen with PSO; it is also much more severe. For example, ACO very rarely

found the correct answer for the 12 CA problem (1 of the 15 scenarios considered), whereas PSO was still able find the correct answer about 40% of the time for the 12 CA case. The GA, however, seemed to consistently find the correct answer regardless of the constraint setting. There is some degradation seen in its performance with increasing dimensionality, but it is still able to find the correct answer about 90% of the time.

A slightly different slice of these same results is provided in Figure 99. Here, a very similar set of histograms are provided, but rather than organizing them by the number of CAs as was done in Figure 98, these are sorted by scenario.

In Figure 99, the various panes are organized such that the 5 front shape – spacing combinations selected in Section 8.4 are represented with the 5 rows. As one moves from the left-most column to the right-most column, the number of choices on a given front increases. From this, one can see how each method does for a given scenario, but, more importantly, one can (relatively) easily see how it does for a given front shape – spacing option as the number of choices increases. Looking at the left-most column of histogram panes, the same basic statements made about Figure 98 can be made: that the GA finds the right answer, the PSO finds the right answer so long as the constraint is at least moderately significant, and the ACO does well unless the constraint is significant.

As the reader moves to the center column of histogram panes, some interesting trends begin to appear. Here, one can see that the PSO performance generally mentioned earlier is only occurring for certain scenarios. For example, in Scenario 10, the bottom center pane, the PSO always finds the right answer, whereas in Scenario 8, the center pane, it does poorly unless the constraint is stringent. One can also see that the GA is not performing well in Scenario 6, only finding the correct answer about 50% of the time for the highly constrained case, and the ACO does not perform well for any highly constrained instances.

Moving finally to the right-most column of histogram panes, the reader sees some interesting trends. The PSO does not perform well for Scenarios 13 regardless of the constraint setting, but it finds the correct answer under all constraint settings all the time for Scenario 15. Interestingly, Scenario 15 was the scenario selected for parameter tuning, which explains why PSO performed so well in that study but does not perform nearly as well for some of

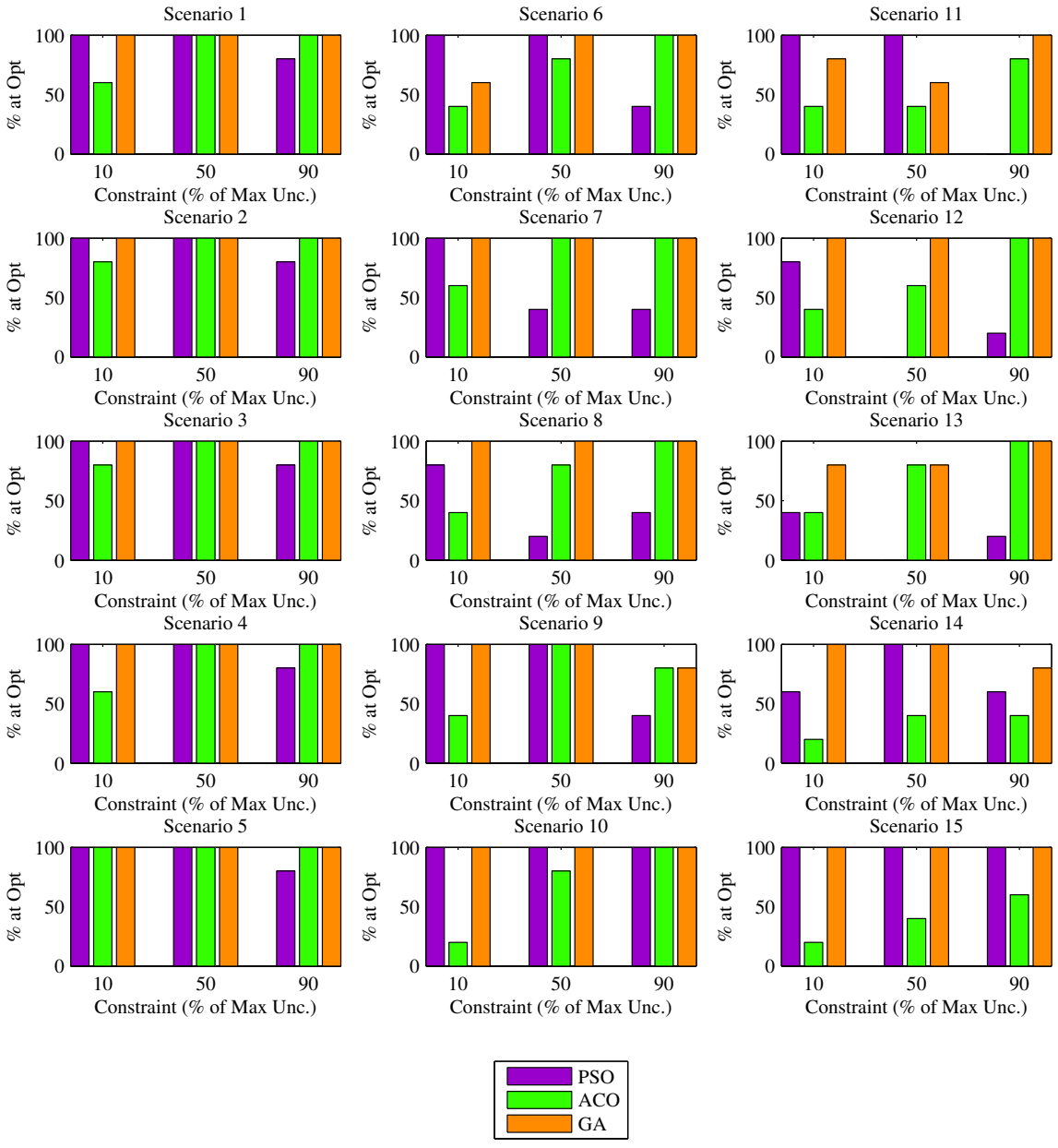


Figure 99: Academic Function FSP Results – Percent of Iterations Where Methods Found the "Optimum" Sorted by Constraint and Scenario.

the other scenarios. Considering the GA performance for the right-most column of panes, it generally does well except in Scenario 11 (interestingly the same front shape – spacing combination as Scenario 6). The ACO also does not perform well for Scenarios 14 or 15, but generally finds the right answer for marginally constrained instances of Scenarios 11 through 13.

From these results, one can see that the scenario examined plays a *large* part in whether or not the method is able to find the correct answer and that the trends seen are a clear function of the number of choices (moving from the left column of panes to the right column of panes). The GA generally finds the correct answer regardless of scenario, while the PSO does very well for some front shape – spacing combinations and the ACO generally underperforms the other two methods considered.

Now that the methods have been examined to determine whether or not they find the optimum, the second important consideration is the variability in their ability to find an answer. To consider this, the various methods' distance from the "optimum" was examined as a function of the number of CAs and constraint setting. This is provided in Figure 100.

In Figure 100, the top pane represents data for a marginally constrained problem, the center pane for a moderately constrained problem, and the bottom pane for a highly constrained problem. The horizontal axis in each pane is the number of inputs for the FSP (or equivalently, the number of CAs). The vertical axis for each pane is the normalized distance from the "optimum". The distance from the "optimum" is normalized by the total size of the space. The reason the distance is normalized is because of the FSP implementation. Each CA has an uncertainty from 0 to 1 and a time cost from 0 to 1. Therefore, the Euclidean time cost/uncertainty "distance" of a given point from the "optimum" will artificially grow as the number of CAs grows just because the total space the point would have to move grows. For this reason, the problem was normalized to provide a more consistent way to view the results. When considering each pane, the results are provided for each method in two separate forms. First, a bar is provided indicating the $\mu \pm \sigma$ for all of the samples taken. However, this information is generally only of use when considering unimodal distributions. Because this is not necessarily the case (Figure 99 shows a strong

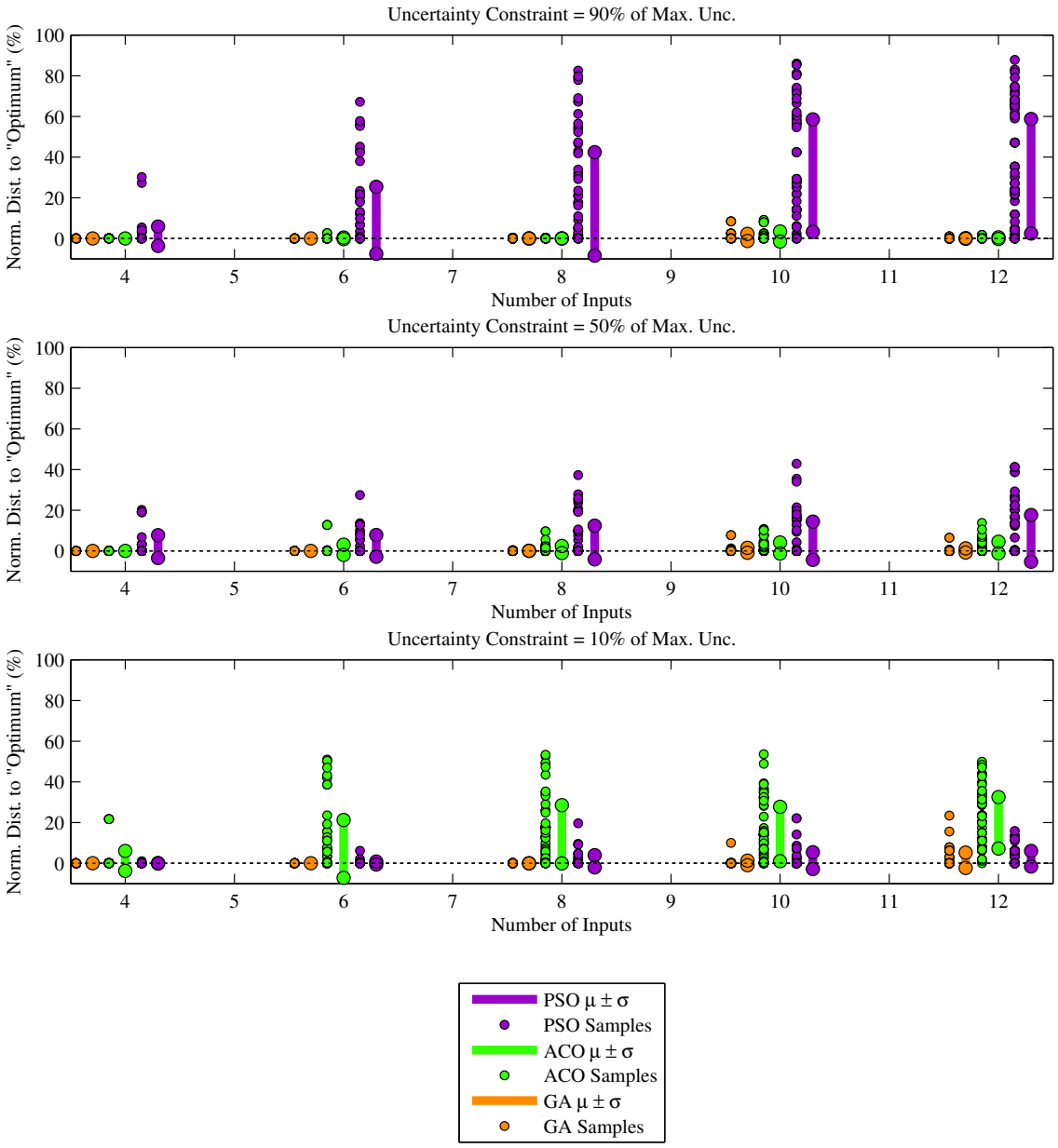


Figure 100: Academic Function FSP Results – Distance from "Optimum" for Various Numbers of CAs.

dependence on the scenario considered), the actual samples are provided just to the left of the bar.

The top pane, representing the marginally constrained situation shows that the PSO has a very large variability (bar) which increases as the number of CAs increases. Examining the samples, one can see that they are well distributed, but there is a clear "clumping" of scenarios that find the right answer (or are close) and those that do not. This reinforces the information provided in Figures 98 and 99. The GA and ACO generally performed well for the marginally constrained problem, and one can see that both the bar and individual samples for both methods are essentially a "dot" on the correct answer.

The middle pane, representing the moderately constrained situation, shows that the PSO has many of the same trends seen in the top pane (a large variability growing with dimensionality), but the variability is much lower. There is still a clear bimodal behavior for PSO, but the magnitude of the variability is much lower. Looking at the ACO, a clear bimodal distribution has formed, though it has a relatively small magnitude, and some variability is also seen in the GA, though it is smaller than that for either of the other two methods. The ACO and GA trends also grow with FSP dimensionality.

Considering the bottom pane, which represents a highly constrained situation, there is a large variability in ACO performance that grows with dimensionality. This performance is clearly bimodal, though for the first time, the ACO $\mu \pm \sigma$ band no longer crosses 0 (the "optimum"). Some variability is seen in GA and PSO performance. The PSO variability is slightly larger, though both are much smaller than that seen for ACO.

Figure 100 provides a more detailed view of information previously shown while taking into account the variability of the system. One can see a relatively large variability in distance from the "optimum" in PSO for marginally constrained problems (though this behavior is bimodal) and a relatively large variability in distance from the "optimum" in ACO for the highly constrained problems. The GA had a comparatively low variability for all constraint settings.

While distance from the "optimum" is an important metric, another is the cost to achieve the result. Because function evaluations are not necessarily cheap when solving the

FSP, the number of function evaluations was examined. This is provided in Figure 101.

Figure 101 is set up much like Figure 100, where each pane corresponds to a different constraint setting and the horizontal axis in each plot is the number of CAs. Much like Figure 100, in Figure 101 the purple corresponds to PSO, the green to ACO, and the orange to GA. The $\mu \pm \sigma$ are shown with a band, and the actual samples are provided. The difference is that the vertical axis in each pane for Figure 101 is the number of function evaluations taken. In this figure, one can see that there is a clear trend in the number of function evaluations with respect to the number of CAs. In this band, one can see that while the PSO band (and samples) do, under certain circumstances, fall below the GA and ACO samples (meaning PSO would require fewer function evaluations), there are also many instances where PSO requires many more function evaluations to converge. This variability increases significantly with the number of CAs and does not seem to be a strong function of the constraint setting. Considering the ACO results, one can see a similar trend. There are instances when the ACO outperforms the GA, but also many instances where it underperforms it as well. The number of function evaluations taken by ACO also seems to vary somewhat with the constraint location. Moving on to the GA results, one can see that it too varies as a function of the number of CAs. Additionally, it has a lower variability with respect to number of function evaluations relative to the other two methods. For this reason, one could say that the GA is a "smart" choice because the answer can be obtained in a relatively predictable number of function evaluations.

One interesting note is that there is a large variability in the function evaluation data. There is a "clumping" which seems to be caused by some parameter not accounted for in this figure. The parameter causing this variation is the scenario number. It was not included in this figure, because, as was stated earlier, one likely does not know what settings of the canonical FSP one has before solving the problem.

In order to better gain insight as to why this "clumping" is occurring, one should consider a scatterplot matrix of the scenario number, the distance from the "optimum", and the number of function evaluations, which is provided in Figure 102. In this figure, the color scheme used is the same as other figures in this chapter, where purple is used

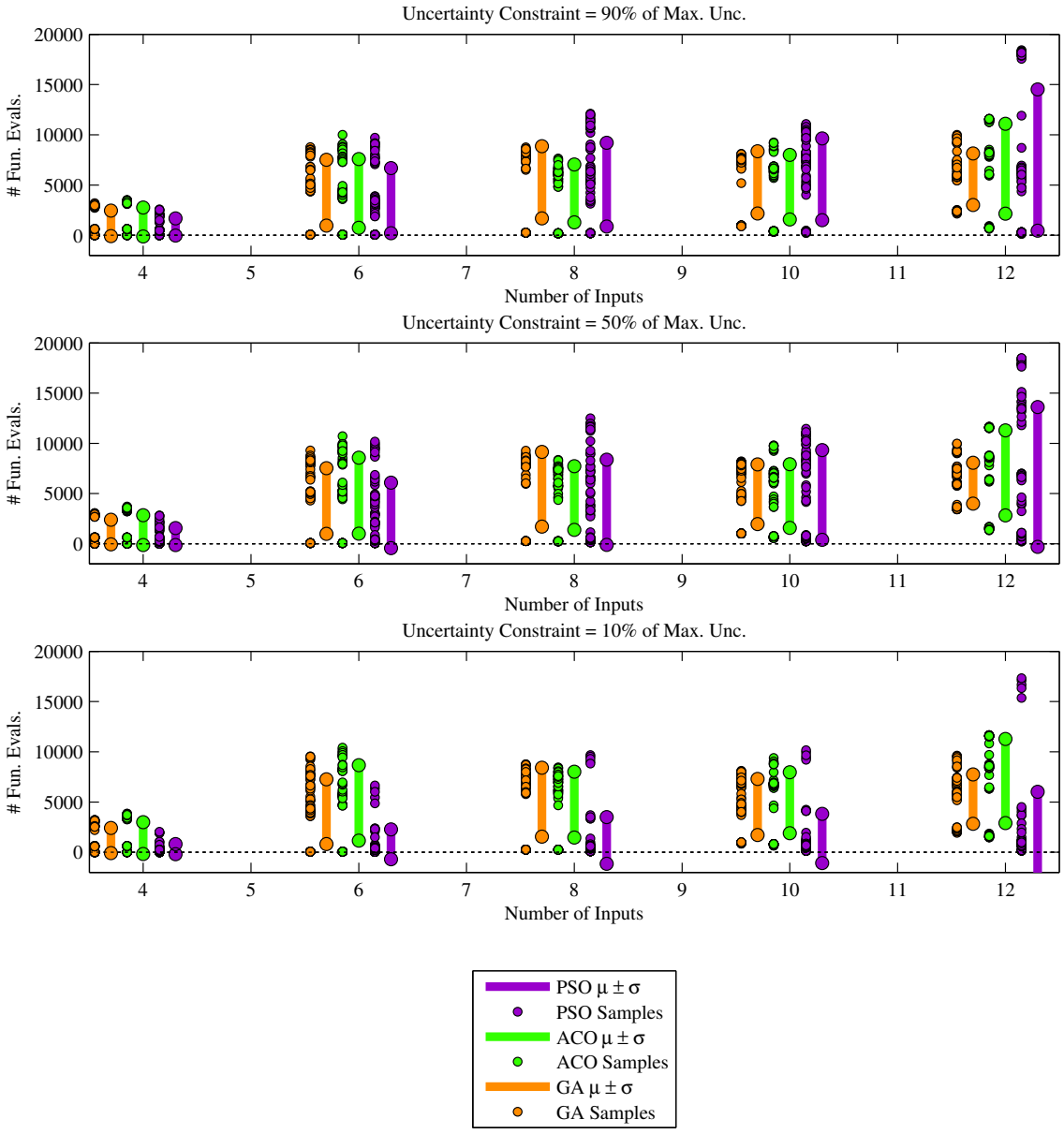


Figure 101: Academic Function FSP Results – Number of Function Evaluations Taken for Various Numbers of CAs.

to represent PSO samples, green is used to represent ACO samples, and orange is used to represent GA samples. When reading a scatterplot matrix, one can see any correlation (or a lack thereof) between any two parameters. This is done by examining the pane found by the intersection of the respective titles. For example, to see the trend between the scenario number and the number of function evaluations, where the scenario number is the vertical axis and the number of function evaluations is the horizontal axis, one should examine the top center pane in Figure 102. Similarly, if one wished to instead view the transpose of this, meaning the scenario number is the horizontal axis and the number of function evaluations is the vertical axis, one should examine the left center pane.

In this figure, one can see a very distinct trend between the number of function evaluations and the scenario number. One may remember from Section 8.4 that the first 5 samples correspond to the 5 front shape – spacing combinations with 2 choices per CA, the middle 5 scenarios samples have 5 choices per CA, and the last 5 have 8 choices per CA. In Figure 102, one can see something resembling "stair steps", where the first 5 scenarios require roughly the same number of function evaluations, the second 5 require roughly the same number, and the last 5 require roughly the same number. From this information, one can now more clearly see that the number of choices, something one has control over when solving the FSP, drives the variability in Figure 101. From this, one could then revise the statement previously made about the "smart choice". This means that the lowest "clump" of samples in Figure 101 corresponds to results from the first 5 scenarios, the next "clump" is from the second 5 scenarios, and the final "clump" corresponds so the final 5. Also, because a user would know a priori how many analysis choices they had for each CA they would be able to better estimate how many function evaluations would be required.

In the bottom-most "clump" for each method in Figure 101, one can see that in some circumstances PSO requires fewer evaluations than ACO and ACO requires fewer evaluations than the GA. The reason for this is because of how the methods fundamentally converge on a solution. The GA does not necessarily converge very quickly; it more thoroughly explores the space. The ACO converges more quickly on a solution, while the PSO converges even more quickly than the ACO. This reinforces discussion provided in Chapter 6. One

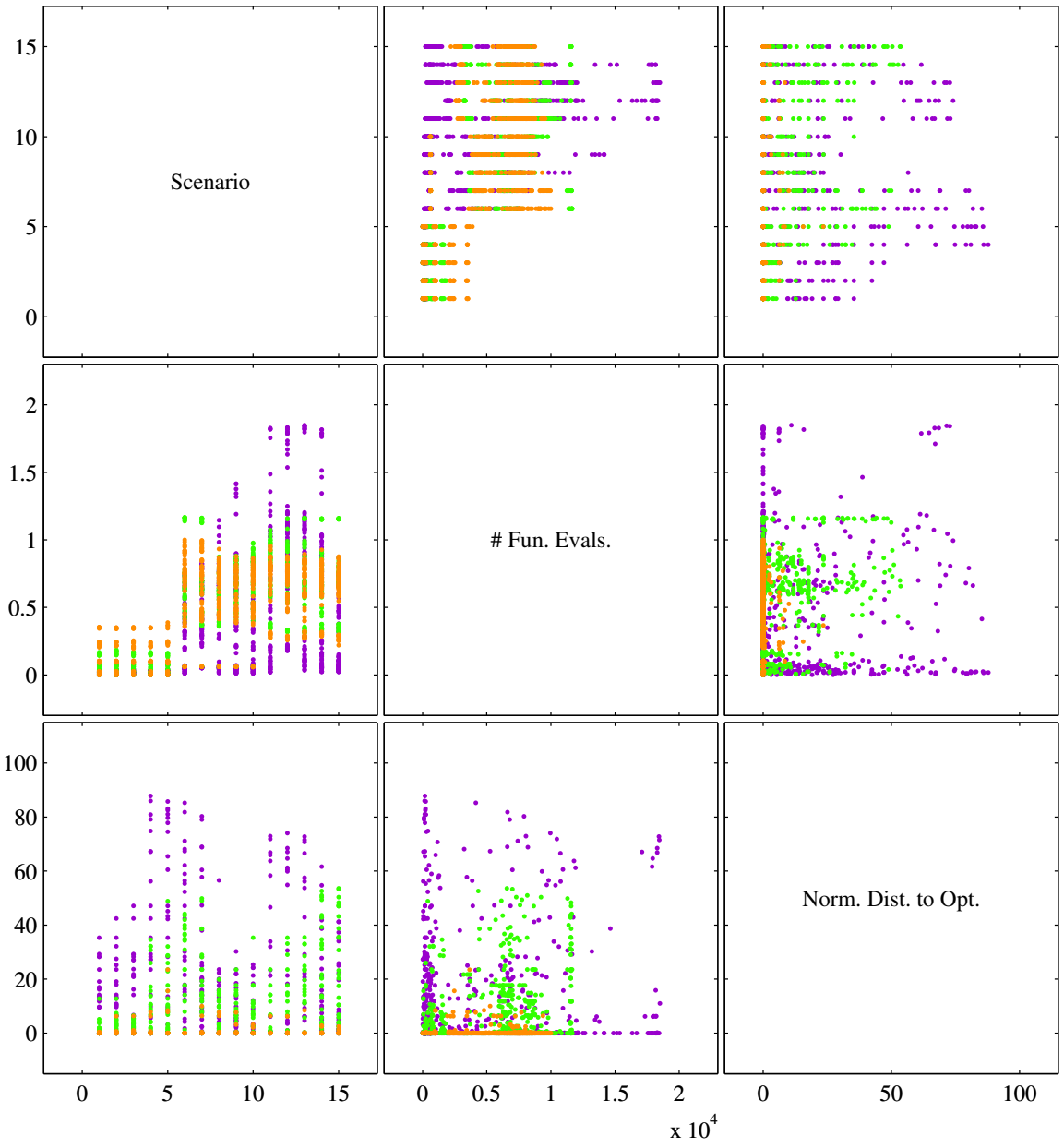


Figure 102: Academic Function FSP Results – Scatterplot Matrix Colored by Model.

important side note is that in all of these situations where PSO requires fewer function evaluations than the other two methods, a relatively "small" problem is being considered. When there are 4 CAs, this means there are only 2^4, or 32, possible combinations, and in all instances, the full factorial was sampled by each method. A difference between the various methods' number of function evaluations only begins to appear once there are at least 10 CAs, or $1,024$ choices. In this situation, the GA, and at times the ACO, are evaluating the full factorial, but the PSO is not. The distinction, which, as stated earlier, was caused by the methods' convergence properties, becomes more distinct as the number of CAs further increases. Therefore, one could say that if there are fewer than 1,000 combinations, evaluating the full factorial is just as effective as employing these methods because they will evaluate the full factorial before converging anyway.

Additionally, if relatively few options exist for each CA but there are 10 or more CAs, PSO will most quickly find a solution. One should notice that this is only *a* solution, not necessarily the *correct* solution. As was shown earlier in Figures 98 and 99, the only method which consistently finds the correct answer, regardless of scenario (not known a priori) or number of CAs, is the GA. Therefore, the GA is still the "best" choice because it consistently finds the correct answer (though under some circumstances it can be obtained more cheaply with other methods). Additionally, if there are $1,000$ or fewer possible choices, one should simply evaluate the full factorial of possible choices to find the correct result.

8.7 Conclusions

In this chapter, a surrogate for the FSP was created and then solved. First, the expected trade-off between the uncertainty in a result and the cost required to get the result was confirmed. This has been abstractly discussed throughout the document but in this chapter actual data was used to confirm its existence. Next, the FSP scope was considered. In order to better test the various methods' ability to find the best tool choice, the analysis being conducted was decoupled from tool selection. This allowed for the characteristics of the uncertainty – time cost trade-off to determine method fitness instead of the analyses available for a specific test. To emphasize this point, a canonical form for the FSP was created. This

was done by identifying a series of FSP characteristics. Given these characteristics, one can create any possible actual tool selection.

The prohibitive size of the FSP led to a scenario-based analysis. Fifteen potentially best- and worst-case scenarios were identified so that one could gain insight about optimizer performance in a computationally feasible way. The three metaheuristic methods considered, GA, ACO, and PSO, are typically used when one assumes function evaluations are relatively "cheap". Because this is not necessarily the case for the FSP because uncertainty quantification may be expensive depending on problem size, a rough algorithmic "tuning" was necessary. Three key inputs for each metaheuristic method were identified and the settings which most consistently found the correct answer in the fewest number of function evaluations were selected. These "tuned" methods were then used to evaluate the 15 scenarios given various constraint settings and problem size. After interpreting these results, it was recommended that if fewer than $1,000$ combinations exist for a given FSP, the full factorial should be evaluated. Otherwise, the user should use the "tuned" GA to find the best tool set.

The next chapter will use this "tuned" GA to select the best tool set for a gas turbine engine analysis. This will involve quantifying uncertainty present in analysis using physical test data and using the Frontier Finding Particle Swarm Optimization (FFPSO) selected in Chapter 6 to quantify uncertainty in a given tool set.

CHAPTER IX

METHOD ILLUSTRATION: GAS TURBINE ENGINE MODELING

9.1 Introduction

The purpose of this chapter is to outline how the process discussed in this document can be implemented for an example. The example problem chosen to illustrate this process is a gas turbine engine model. Physical test data was taken from the National Aeronautics and Space Administration (NASA) Energy Efficient Engine (E^3) experiments conducted in the 1970's and 1980's [210] to quantify the uncertainty in a series of thermal analysis tools which were used for gas turbine engine modeling. Once the uncertainty in these tools was quantified, the trade-off between model uncertainty and computational cost was simulated. The simulated uncertainty in a selection of analyses was quantified using Frontier Finding Particle Swarm Optimization (FFPSO) described in Chapter 6. The Genetic Algorithm (GA) described in Chapter 8 was then used to select which analysis combinations would most effectively produce results of a necessary accuracy while minimizing the amount of time used to obtain those results. This was completed for each of the 15 scenarios discussed in Chapter 8.

9.2 Available Data for Uncertainty Quantification

In the 1970's and 1980's NASA conducted a series of tests with General Electric (GE) and Pratt and Whitney (P&W) on developing an E^3. The data from these tests served as a test-bed for several technologies in the next generation of aircraft engines, specifically the GE90, $CF6-80$, and PW4000.[210] The data from these tests was then published as a series of NASA technical reports on the various components tested, including the High Pressure Compressor (HPC)[107], the High Pressure Turbine (HPT)[208], the core (all components on the high pressure spool)[66], and the integration of the low pressure spool and the high pressure spool[67]. This data is publicly available and was used to quantify the uncertainty in a series of tools used for thermal analysis of gas turbine engines.

The data used for this example was extracted from a final report and deliverable

submitted by the Aerospace Systems Design Laboratory (ASDL) for the University Research, Engineering, and Technology Institutes (URETI) for Aeropropulsion and Power Technology (UAPT). In this contract, a multi-fideltiy Modeling and Simulation (M&S) environment, Virtual Integrated Propulsion Environment for Revolutionary Concepts, Architectures, and Technologies (VIPER-CAT) was created in order to assess the impact of potential technologies for gas turbine engines.[2] In these experiments, a series of tools was integrated in an Numerical Propulsion System Simulation (NPSS) environment to conduct gas turbine engine analysis. As each tool was added, its uncertainty relative to the available E^3 data was determined. This environment was initially considered as an example problem for this fidelity selection method, but due to the lack of depth in this environment, the results provided in Chapter 8 would show that the user should simply evaluate the full factorial of possible combinations. Instead, a single set of gas turbine engine analysis tools from the VIPER-CAT environment was used to evaluate a gas turbine engine, and the uncertainty – time cost trade-off, which was confirmed in Chapter 8, was simulated.

9.2.1 Gas Turbine Engine Operation and Analysis

When conducting gas turbine engine analysis, especially off-design analysis, one typically uses a component map to represent the component's performance at a variety of conditions. These maps are often created using nondimensional or corrected parameters. The nondimensionality and corrected parameter properties allow a single map to provide performance information for the component at a wide variety of possible operating conditions. An example of one such component map for a compressor is provided in Figure 103. In this figure, five parameters are represented: corrected speed ($\frac{N}{\sqrt{\theta}}$), corrected flow ($\frac{W\sqrt{\theta}}{\delta}$), compressor pressure ratio, adiabatic efficiency, and R-line. The corrected flow and speed have parameters θ, which is the corrected temperature ($\frac{T}{T_{ref}}$), and δ, which is the corrected pressure ($\frac{P}{P_{ref}}$). The reference temperature and pressure are generally taken to be Sea-Level Static (SLS) conditions. By normalizing the pressure and temperature (and then speed and flow), one is able to reduce the compressor's performance to a single map regardless of operating conditions.

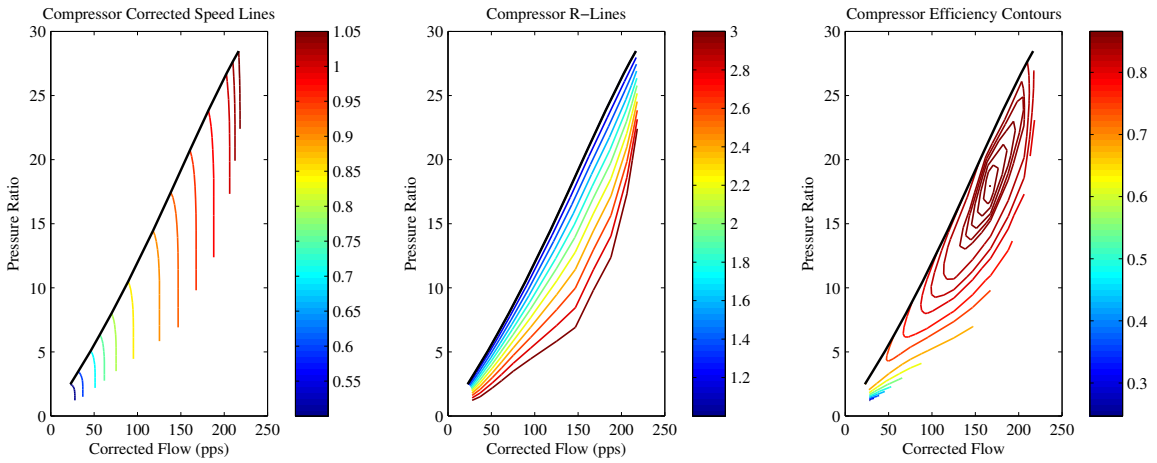


Figure 103: Example Compressor Component Map.

All three panes in Figure 103 have the corrected flow on the horizontal axis and compressor pressure ratio on the vertical axis. The left-most pane contains a plot of the corrected speed lines. In this pane, one can see how pressure ratio and corrected flow vary with corrected speed for the compressor. At the left end of each speed is a thick black line; this represents the surge line. To better understand the surge line, one must examine the components in an axial compressor. Axial compressors, which this figure represents, are made up of a series of disks of small blades which behave much like the wings on an aircraft. When a wing has too high of an angle of attack, meaning the wing is attempting to get more lift than is possible for the flow velocity, it will stall. When a wings stalls, the flow separates and the wing no longer produces lift. Surge occurs when the compressor is attempting to produce more compression (lift) than it can for a given corrected speed (flow velocity).

The center pane contains a plot of the adiabatic efficiency for the compressor. Adiabatic efficiency is a measure of how efficiently the flow is compressed. In an ideal compressor, the efficiency is 100%. Real components have some losses due to entropy increase; and this is measured with the adiabatic efficiency. The right-most plane is that of the R-line, or reference line. These references lines are lines that are artificially created. While they are only a reference, all maps used for this study are calibrated where the surge line (shown in black in all three panes) occurs at R-line = 1.0. Similarly, the compressor design point is at R-line = 2.0. For more information on how map correction is conducted or how these

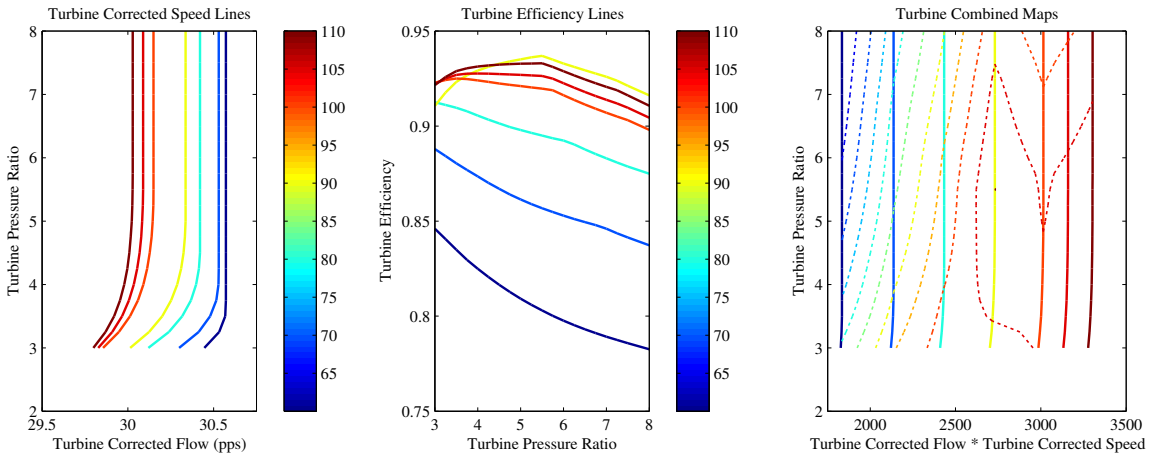


Figure 104: Example Turbine Component Map.

component maps are then used to conduct cycle analysis, the reader is encouraged to consult several texts in the area.[44, 104, 144]

A similar map is used for turbine analysis. Turbine maps contain four parameters: corrected speed, corrected flow, turbine pressure ratio, and adiabatic efficiency. The turbine maps are handled in much the same way as the compressor maps. Example turbine component maps are provided in Figure 104. The left-most pane in Figure 104 has turbine corrected flow provided on the horizontal axis and pressure ratio on the vertical axis. In this pane, the reader can see the corrected speed lines. One interesting point is that in this figure, pressure ratio and speed are independent for most pressure ratios (this was not the case for the compressor map); the reason for this is that for most turbine operation, the turbine is choked. For more information on this phenomena the reader should consult texts on the field.[113] The center pane is a plot of turbine adiabatic efficiency vs. pressure ratio. This efficiency is the same one as that plotted for the compressor. One should note that for this example, as with most compressors and turbines, the turbine adiabatic efficiency is higher than that of the compressor. This occurs because the turbine has a favorable pressure gradient, meaning it is easier to efficiently expand the flow. Conversely, compressors have an adverse pressure gradient, meaning it is more difficult to efficiently compress the flow. These two panes are typically used for turbine analysis. One alternative, discussed in some of the literature, is to combine these two panes into a single plot[144]; this is provided

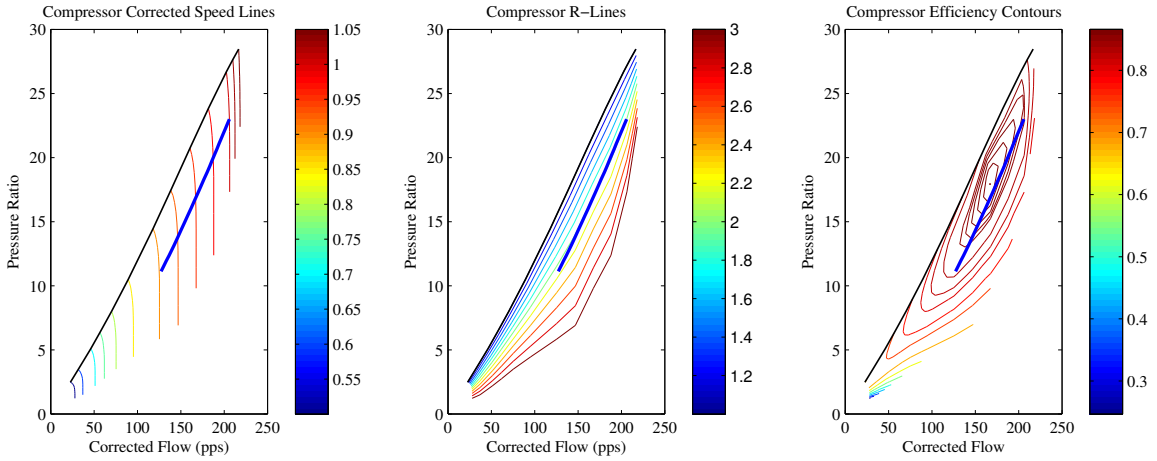


Figure 105: Example Compressor Component Map and Op-Line.

in the right-most pane. In this pane, the speed lines look much like they do in the left-most pane, but are nearly linear at low turbine pressure ratios (when not choked). Additionally, one can see efficiency "islands" behind the speed lines. For more information on how turbine maps are created and used, the reader is encouraged to consult several texts in the area.[44, 104, 144]

When using component maps, such as those illustrated in Figures 103 and 104, an operation line can be calculated. For 0-D analysis, this is based on conservation of mass and energy, and is a "line" on the compressor and turbine maps where the engine operates regardless of flight setting. An illustration of an operation line, or "op-line", is provided on a compressor map in Figure 105. In this figure, the op-line is shown with a solid blue line. This is the only part of the map that the compressor can operate in due to mass and energy constraints from the other components in the cycle.

If one were to quantify the uncertainty in a map, it would make sense to calculate how well a given method predicts the engine's performance along an op-line. In these regions, the flow can be best predicted and it is likely that a given method's uncertainty in this region would be most representative of how it predicts the engine's overall performance. However, as was mentioned earlier, the op-line for a given component is due to mass and energy constraints from the other components. Therefore, it is very likely that if one component being modeled had a significantly different performance relative to some actual component,

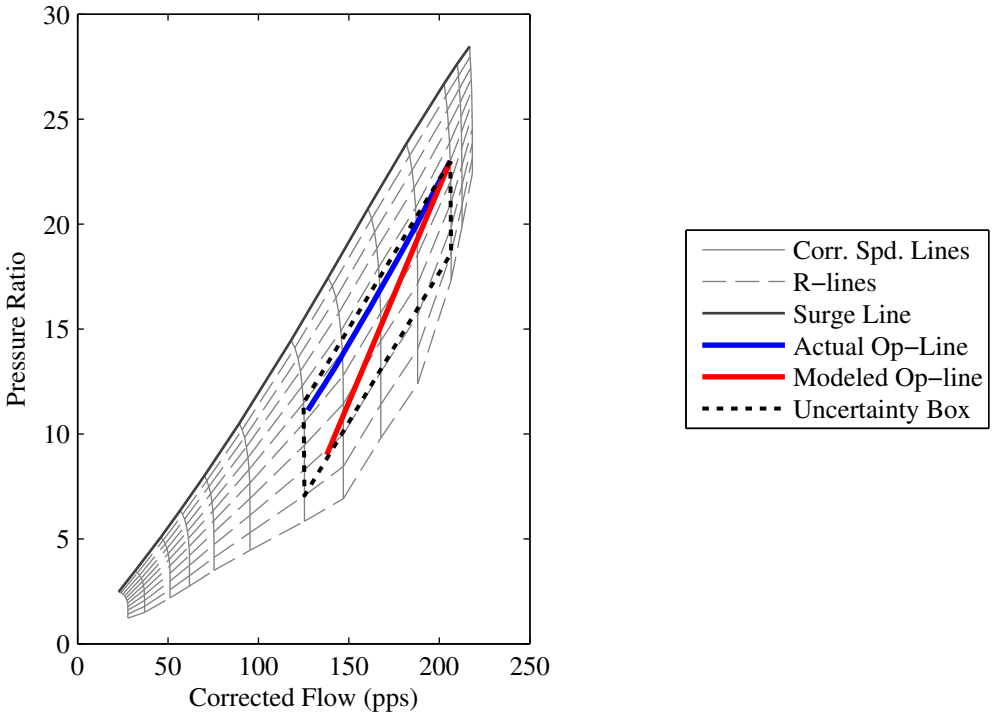


Figure 106: Illustration of Uncertainty Quantification for Compressor Maps Using Op-Lines.

the op-line would move significantly. Therefore, the following approach was taken: the op-line for the actual test, obtained from physical test data [2], was calculated. Then, the op-line for the test configuration, as modeled with selected analysis tools was calculated. These op-lines can then be compared and a corrected speed/R-line uncertainty "box" can be calculated. Within this box, the uncertainty in the model relative to the actual test data can be determined. This process and an example "uncertainty box" are provided in Figure 106. Once the uncertainty in available map parameters has been identified, the FFPSO can be used to quantify the uncertainty in any output or outputs of interest from the cycle.

This process was implemented for available test data for the E^3 fan, E^3 HPC, and E^3 HPT to quantify uncertainty in three available tools. The uncertainty quantified in each will be discussed in more detail in subsequent subsections.

9.2.2 Example Problem Engine Architecture

Given that the truth data was from the E^3 study, an engine architecture similar to that was selected. For that reason, a 300-passenger-class engine model was created. This engine was intended to be similar to the GE90 made by GE and the PW4090 made by P&W. This engine is a similar thrust class to the Trent 800 created by Rolls-Royce, but is a 2-spool engine not a 3-spool engine like the Trent 800.

This engine for this example problem was modeled in a software package developed by NASA to conduct system-level engine modeling: NPSS. NPSS is currently maintained by the NPSS consortium, which is composed of representatives from all major US engine manufacturers. This software has been used in a wide variety of studies for academia[143, 178, 185], industry[75, 187], and government [32, 138] work.

A schematic for the engine architecture being used to quantify the uncertainty in the map generation tools is provided in Figure 107. As shown in the figure, this is a 2-spool Separate Flow Turbofan (SFTF) engine. There are three compressors (fan, booster, and HPC) and two turbines (HPT and Low Pressure Turbine (LPT)). The LPT is driving the fan and booster, while the HPT is driving the HPC. The design point was taken to be a cruise point in some notional mission (altitude = 35,000 ft, Mach 0.8). Throughout this figure, a series of flow stations are labeled with numbers. The numbering scheme used was taken from the **SAE!** (**SAE!**) Aerospace Recommended Practices 755.[1] For more information, the reader is encouraged to consult this reference.

This cycle was evaluated with two different sets of maps. The first was with maps created using the map generation tools which will be described in subsequent sections. The second cycle was evaluated with E^3 maps, which will also be discussed in subsequent sections. From these two evaluations, the op-lines for both the actual physical tests (E^3 maps) and from the analysis of interest (generated maps) were determined and the uncertainty in the maps in the corrected speed/R-line "box", as discussed earlier, was calculated.

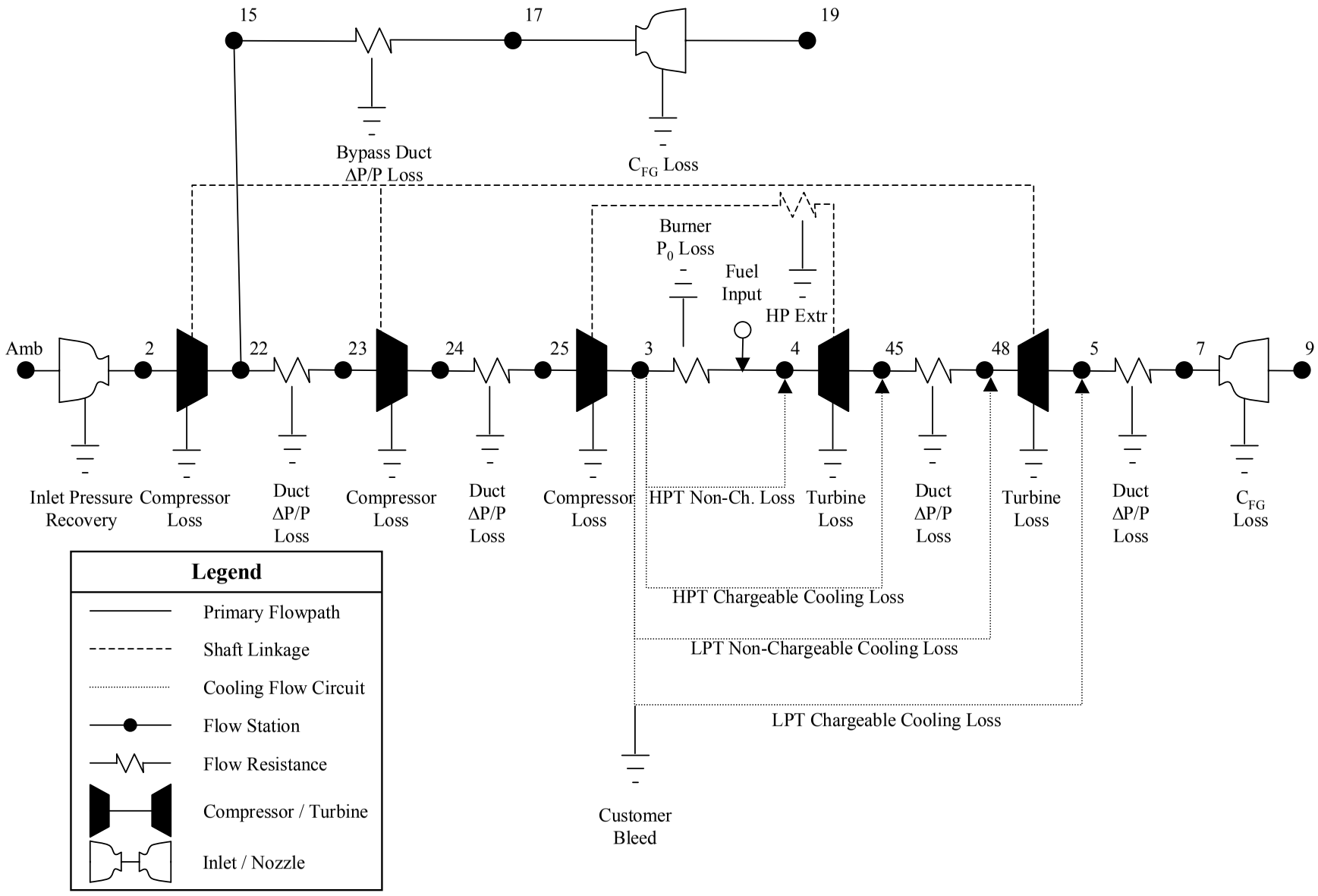


Figure 107: Example Problem SFTF Engine Architecture.

9.2.3 Fan Energy Efficient Engine Data, Analysis Tool, and Uncertainty Quantification

An E^3 fan performance map was obtained from the University Research, Engineering, and Technology Institutes (URETI) final deliverable[2] that was based on the E^3 performance data[67]. This map was considered, for the purposes of this example, to be the actual E^3 performance data for the fan.

A program called Flow Modulating Fan (MODFAN) was used to create the fan map. MODFAN was developed in the early 1980's by GE to parametrically create off-design maps for both axial and centrifugal fans.[40] This computer program was developed as a more accurate alternative to taking a generic fan map and scaling it by flow, pressure ratio, and efficiency at the design point. For more information on how this scaling could be conducted, one should consult texts in the field.[144, 44] Instead of simply scaling a generic map, MODFAN has a parametric "backbone" resembling an Response Surface Equation (RSE) which is scaled. This program has been used with success in creating maps for a variety of products, including the Environmental Design Space (EDS) [200] which was created for the Federal Aviation Administration (FAA) to enhance the environmental policy making process[120], and the URETI VIPER-CAT multi-fidelity environment for NASA.[2] The fan map created using MODFAN given all E^3 specifications is provided in the right column of panes in Figure 108 for reference.

The left column of panes in this figure contain what is being considered the E^3 fan map. The top row of panes contain the corrected speed lines (N_C) for the fan maps. The center row contains efficiency contours, and the bottom row of panes contain the R-line plots for the maps. While these are arbitrary reference lines, they have been set up such that the design point is at R-line = 2.0 and the surge line (discussed in the previous section) is at R-line = 1.0. From these plots, one can see that while the maps are similar, there are significant differences, specifically that the corrected speed lines have a much more gradual slope at high R-line values for the E^3 map, the efficiency contours are much less "smooth" for the E^3 map, and the R-lines are also much less smooth.

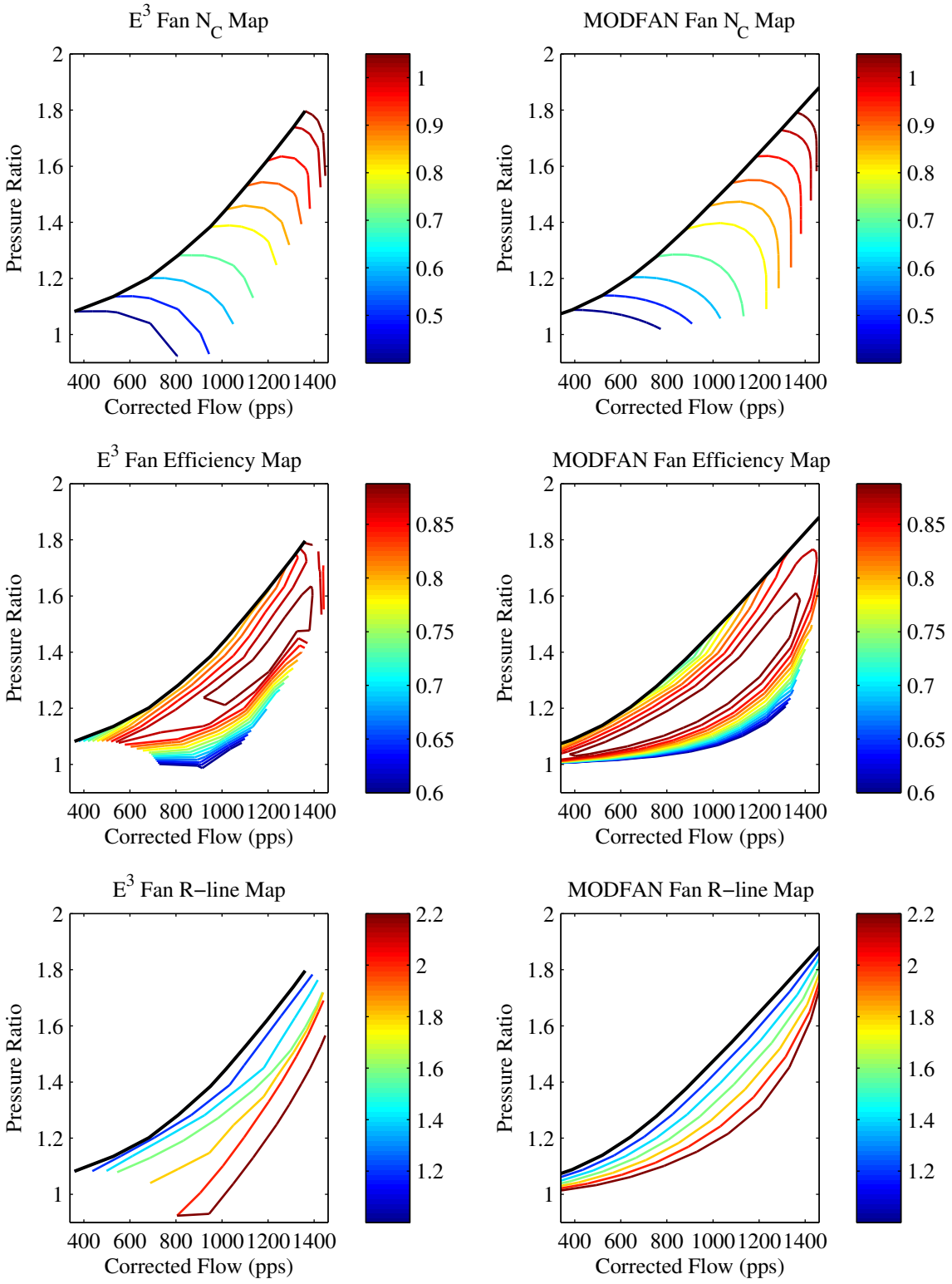


Figure 108: Comparison of E^3 and MODFAN Fan Maps.

Table 11: Maximum and Minimum R-line and Corrected Speed Values for Fan Powerhook.

Model	Max R-line	Min R-line	Max N_C	Min N_C
E^3 Fan	1.57	1.53	0.81	0.44
MODFAN	1.75	1.44	0.93	0.46
Uncertainty Box	1.75	1.44	0.93	0.44

Table 12: Maximum and Minimum Uncertainty in Efficiency and Corrected Flow for MODFAN.

Parameter	Max Difference	Min Difference	Max % Difference	Min % Difference
Efficiency	0.039	$-.0475$	4.47%	-5.83%
W_C (pps)	62.35	-8.52	5.28%	-1.39%

Once MODFAN had been used to create the fan map, the uncertainty could be quantified. This was done using the op-line process discussed earlier and illustrated in Figure 106. After running a powerhook at Sea-Level Static (SLS), the maximum and minimum R-line and corrected speed values were obtained. These are provided in Table 11.

Given this uncertainty, the percent difference in maps could be calculated. This was performed for both the efficiency and corrected flow maps. While a third parameter could have been created for the fan pressure ratio, it was not due to modeling challenges in the chosen simulation software. The uncertainty bounds for the efficiency and corrected flow (W_C) are provided in Table 12. The actual differences are provided in the center, and the percent differences are provided on the left in this table. The reason for this is that when implemented for some new model, the percent differences, rather than the actual differences, should be applied to the new model around the design point. This would allow for one to use uncertainty data obtained for the E^3 configuration much like a "coefficient", so that it can be scaled up or down to some new model.

9.2.4 High Pressure Compressor Energy Efficient Engine Data, Analysis Tool, and Uncertainty Quantification

An E^3 HPC map was obtained from the URETI final deliverable[2] that was based on the E^3 performance map data found in the final E^3 performance documents [107, 66]. This URETI map was considered, for the purposes of this illustration, to be actual E^3 performance data for the HPC.

A program called Compressor Map Generator (CMGEN) was used to create the HPC map. CMGEN was developed in the early 1980's by GE to parametrically create off-design maps for axial and centrifugal compressors.[39] This program is something of a "cousin" to MODFAN (discussed earlier), as they were created at the same time (early 1980's) at the same place (GE) for the same customer (NASA). These programs also have a very similar approachs to the problem; CMGEN also uses an RSE-like "backbone" to parametrically create these maps. It, like MODFAN, was shown to better approximate test data than the previously used map-scaling procedure.[39] CMGEN has been used by many of the same applications as MODFAN, including the EDS environment for the FAA[120, 200] and the URETI VIPER-CAT environment for NASA[2]. The CMGEN HPC map is compared against the E^3 original in Figure 109 for reference.

In this figure, the E^3 map is provided in the left column of panes and the CMGEN map is provided in the right column of panes. The top row of panes contains the corrected speed maps. The center row of panes contains the efficiency maps, and the bottom row of panes contains the R-line maps. As was the case with the fan maps, the R-lines have been scaled such that R-line = 1.0 is the surge line and R-line = 2.0 is the design point. Comparing the left and right columns of panes, one can see that they are similar but some clear differences can also be seen. For example, the CMGEN results show that for the same R-line, a lower pressure ratio and higher speed can be obtained (speed lines are longer). The efficiency contours for the E^3 data seem to be more smooth, whereas the results for CMGEN seem much less smooth.

Using the op-lines process discussed earlier, the uncertainty in the CMGEN map was determined. The resulting R-line/corrected speed uncertainty "box" is provided in Table

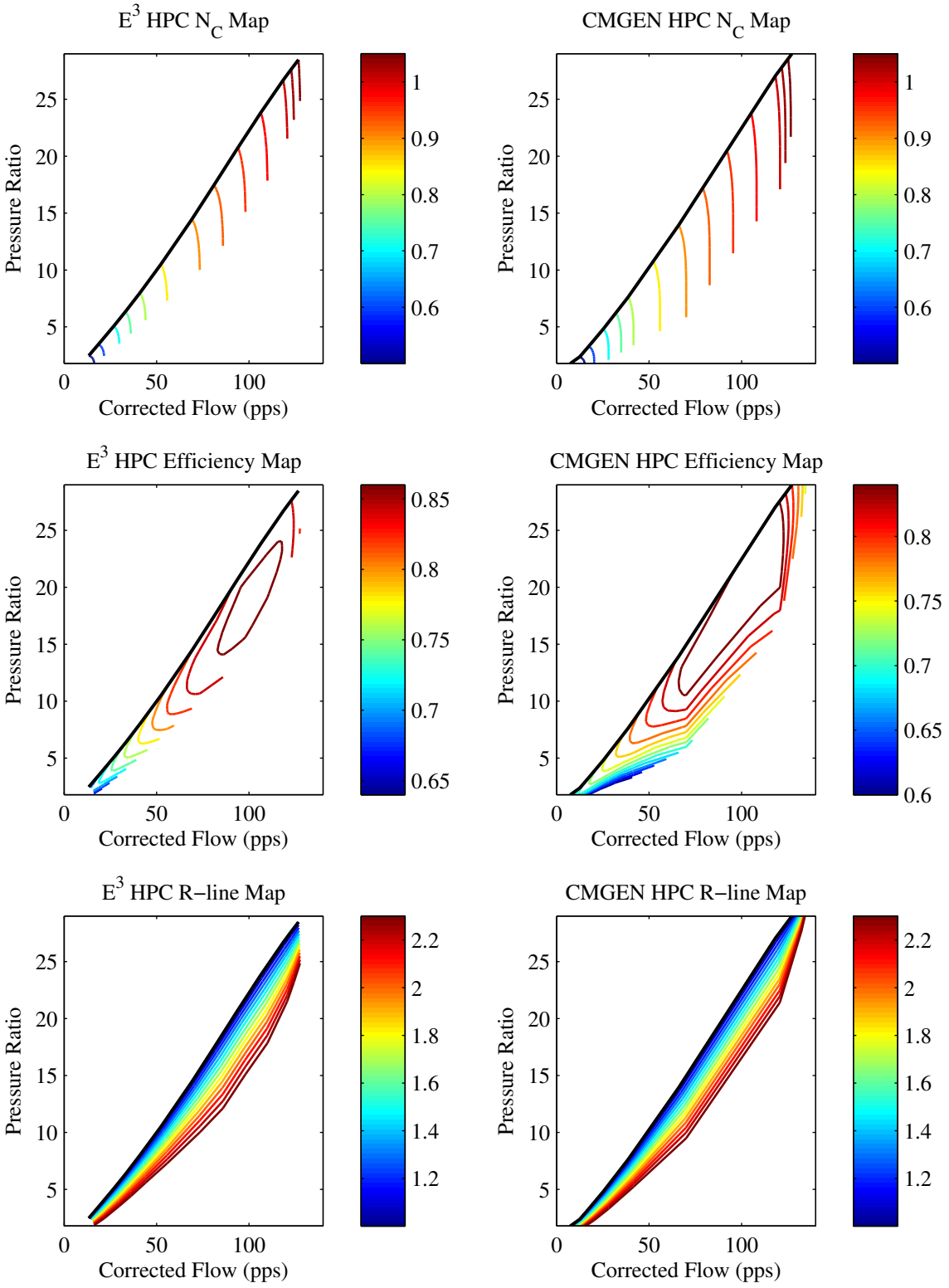


Figure 109: Comparison of E^3 and CMGEN HPC Maps.

Table 13: Uncertainty Box and Percent Maximum and Minimum R-line and Corrected Speed Values for HPC Powerhook.

Model	Max R-line	Min R-line	Max N_C	Min N_C
E^3 HPC	2.03	1.93	0.976	0.865
CMGEN	2.03	1.95	0.971	0.871
Uncertainty Box	2.03	1.93	0.976	0.865

Table 14: Maximum and Minimum Uncertainty in Efficiency and Corrected Flow for CMGEN.

Parameter	Max Difference	Min Difference	Max % Difference	Min % Difference
Efficiency	0.009	0	1.01%	0%
W_C (pps)	3.85	−0.35	4.53%	−0.33%

13.

Once the uncertainty "box" was obtained, the maximum differences in the adiabatic efficiency and corrected flow were found. These values are provided in Table 14. As was the case with the fan, the uncertainty in the pressure ratio could have been determined, but was not due to modeling difficulties in the NPSS model. As was the case with the MODFAN data provided earlier, this CMGEN data table contains both actual and percent differences. The reason for this is so that the uncertainty obtained for this specific E^3 configuration and size (discussed in an earlier subsection) can be applied to a new, similar model more appropriately.

9.2.5 High Pressure Turbine Energy Efficient Engine Data, Analysis Tool, and Uncertainty Quantification

The E^3 HPT performance map was obtained from the URETI final deliverable.[2] This map was derived from the E^3 performance documents published in the early 1980's by NASA and GE[66, 208], and for the purposes of this illustration, it will be assumed that this HPT map is the actual E^3 performance data.

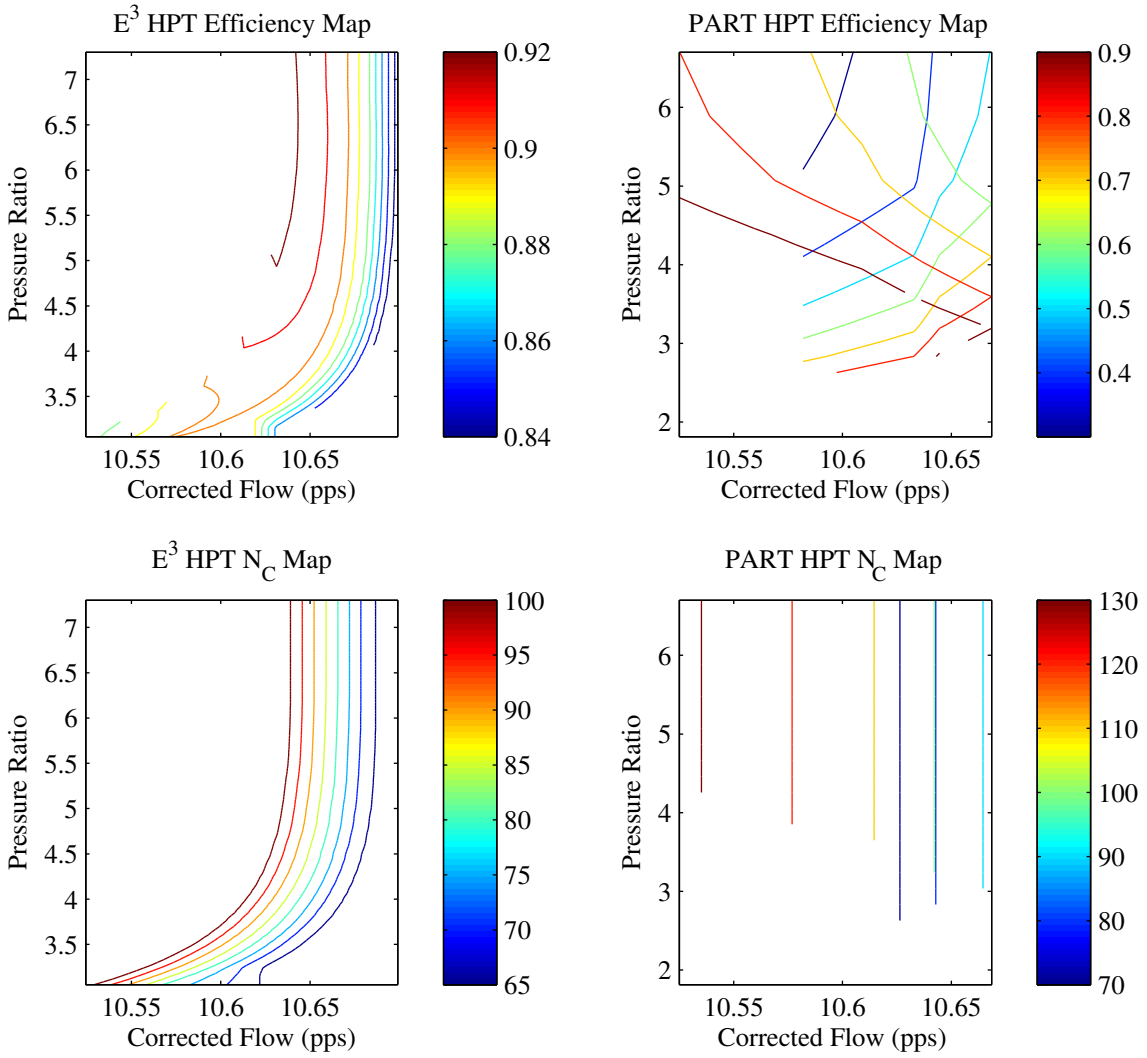


Figure 110: Comparison of E^3 and PART HPT Maps.

The third analysis code considered was Parametric Turbine (PART). PART was created by GE in the early 1980's for NASA, much like MODFAN and CMGEN.[38] These three programs were all part of the same final document and are each a volume. PART operates much like CMGEN and MODFAN in that it has an RSE-like "backbone" which is then scaled and used to parametrically create component maps.[38]. PART, like MODFAN and CMGEN, has been used with success for a wide variety of studies, including the EDS, an environment created for the FAA[120, 200], and the URETI VIPER-CAT environment for NASA[2]. The E^3 and PART turbine performance maps are provided for reference in Figure 110.

Table 15: Maximum and Minimum Pressure Ratio and Corrected Speed Values for HPT Powerhook.

Model	Max PR	Min PR	Max N_C	Min N_C
E^3 HPT	5.74	5.54	108.4	101.4
PART	5.82	5.43	109.6	102.6
Uncertainty Box	5.82	5.43	109.6	101.4

In this figure, the left column of panes is the E^3 map, and the right column of panes is the PART map. The top row of panes is the efficiency map, and the bottom row of panes is the corrected speed map. Like the compressor maps provided earlier in Figures 108 and 109, the turbine maps have been plotted as corrected flow on the horizontal axis and pressure ratio on the vertical axis. In examining the data in this figure, one can see there are several very clear differences between the E^3 and PART data. First, PART doesn't seem to predict un-choked turbine performance; all speed lines are vertical in the bottom right pane. The E^3 data shows some un-choked turbine performance; in the bottom left pane the speed lines curve to the left at lower pressure ratios. Another clear difference can be seen in the efficiency map. The E^3 has relatively straight lines of efficiency, though there seems to be some sort of island forming in the top center portion of the E^3 efficiency plot. The PART data shows no clear trend between efficiency and either pressure ratio or corrected flow. The lines also seem to be crossing each other. This does not make physical sense, and may be due to some internal errors within this version of the software. For the purposes of this exercise, it will be used, but this will likely produce a *significant* uncertainty in the efficiency for PART.

When calculating the uncertainty "box" for the turbine, a slightly different approach must be taken. The R-line and corrected speed were used to create the box for the compressors, but there is no R-line for the turbine. Instead, pressure ratio (PR) was used in lieu of R-line. The resulting pressure ratio/corrected speed uncertainty "box" for PART is provided in Table 15.

After the uncertainty "box" provided in Table 15 was identified, the uncertainty in the

Table 16: Maximum and Minimum Uncertainty in Efficiency and Corrected Flow for PART.

Parameter	Max Difference	Min Difference	Max % Difference	Min % Difference
Efficiency	0.26	0	28.46%	0%
W_C (pps)	0	−0.01	0%	−0.07%

PART map relative to the E^3 map could be calculated. The resulting maximum uncertainties in efficiency and corrected flow are provided in Table 16. As was mentioned earlier, there is a *significant* uncertainty in the turbine adiabatic efficiency. Interesting, there is a very low uncertainty in the turbine corrected flow. This is likely due to the fact that the turbine was choked for the entire powerhook. This means that while PART seems to have not provided any data for the un-choked possibility, it was not significant; (this is an approach taken many times in first order gas turbine analysis[44, 144].

9.3 Experimental Setup

For this method illustration, a 2-spool SFTF engine was used. This engine schematic is the same as the cycle used to model the E^3 engine for uncertainty quantification purposes, but some differences in cycle parameters and operating conditions were made. The design point was taken to be SLS, hot day ($+27°F$); the design point cycle parameters are provided in Table 17.

The cycle was run for a powerhook at SLS, standard day, and the uncertainty in thrust points along this powerhook were determined. For this illustration, two different thrust points were chosen: one at maximum thrust, Power Code (PC)=50, and one at minimum thrust, PC=26. This allows for uncertainty in multiple outputs to be examined rather than in a single output, as was done in the "canonical" FSP in the previous chapter. These outputs are both thrust points, so one might expect them initially to have the same significant factors. While this is generally the case, at lower power settings, the thrust contribution from the fan decreases more rapidly than that from the cycle core. Therefore,

Table 17: Design Point Properties for Example Problem SFTF Engine.

Component	Property	Value	Units
Inlet	Pressure Recovery	0.995	
Fan	Pressure Ratio	1.50	
	$\eta_{adiabatic}$	0.8914	
	Flow Rate	3,100	$\frac{lb_m}{sec}$
Splitter	Bypass Ratio	8.502	
Booster	Pressure Ratio	1.30	
	$\eta_{adiabatic}$	0.8992	
	Flow Rate	326.25	$\frac{lb_m}{sec}$
HPC	Pressure Ratio	20.00	
	$\eta_{adiabatic}$	0.8650	
	Flow Rate	304.43	$\frac{lb_m}{sec}$
Burner	Exit Temperature	3500	$^\circ R$
HPT	Pressure Ratio	5.30	
	$\eta_{adiabatic}$	0.9200	
	Flow Rate	311.47	$\frac{lb_m}{sec}$
LPT	Pressure Ratio	5.29	
	$\eta_{adiabatic}$	0.9300	
	Flow Rate	329.36	$\frac{lb_m}{sec}$
Bypass Nozzle	Pressure Ratio	1.47	
	C_{FG}	0.9975	
	Thrust	75701.1	lb_f
Core Nozzle	Pressure Ratio	1.27	
	C_{FG}	0.9999	
	Thrust	11021.7	lb_f

by considering the maximum and minimum thrust settings, one can see the difference and determine whether or not these metrics are truly 1-to-1.

As was discussed in Subsections 9.2.3, 9.2.4, and 9.2.5, two variables were identified for each compressor and turbine: a corrected flow and adiabatic efficiency. The uncertainty with respect to these was quantified in the various subsections and can be used in the example problem model. While uncertainty information was only obtained for three components, a fan, HPC, and HPT, it will be used for five different components in the example problem. The booster's uncertainty will also be quantified. While it is neither an HPC nor a fan, some argument could be made that it would more closely resemble how an HPC might operate. While this is true, the fan uncertainty information was used instead. The reason for making this selection is because it has a larger uncertainty and is therefore more of a "worst case" approach to the problem. Uncertainty information was not available for the Low Pressure Turbine (LPT), so the uncertainty percentages obtained for the HPT will be used.

9.3.1 Example Problem Uncertainty – Cost Trade-off

In order to show the trade-off between cost and uncertainty, it would be ideal for a series of different tools to be available for a given analysis. While this is present in many instances, it is not available for the purposes of this study. A candidate environment was considered, the URETI VIPER-CAT environment.[2] However, this environment was intended for conceptual and pre-conceptual design. Additionally, there were only around 45 tool combinations available to do the relatively low number of analysis choices for each engine component. For this reason, the somewhat fictionalized approach will be taken.

Rather than actually developing various analysis tools to construct the model uncertainty – time cost trade-off, which was illustrated for a turbojet powerhook earlier in Section 8.2, the trend will be created with the single tool option for each component, which was discussed earlier in this chapter. This single tool and its corresponding uncertainty will be used to artificially create the trade-off by decreasing the uncertainty and applying an "adder" to the result. This is illustrated graphically for a notional compressor analysis in

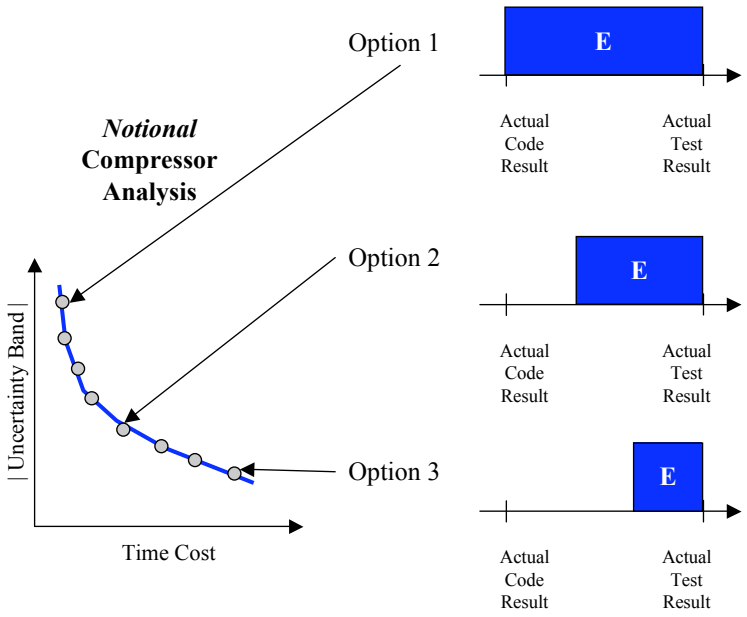


Figure 111: Illustration of Fictionalized Uncertainty – Time Cost Trade-off for Example Problem.

Figure 111.

In this figure, "Option 1" corresponds to the actual uncertainty in this tool. A number of fictional options can then be created to form the uncertainty – time cost trade-off; "Option 1" is the lowest fidelity choice for a given Contributing Analysis (CA). Similarly, "Option 3" is the highest fidelity option, which will always have 5% of the uncertainty from "Option 1". The number of "code" choices and the shape of the uncertainty – time cost front will be determined using the 15 scenarios considered earlier in Chapter 8.

9.3.2 Example Problem Sensitivity Analysis

In order to quantify uncertainty in thrust at these two flight conditions (PC=50 and PC=26), a sensitivity analysis was first conducted. This was done to identify the output from each component which, given the ranges identified in Subsections 9.2.3, 9.2.4, and 9.2.5, has the largest impact on the size of the uncertainty band for each of these two responses.

Because there is the possibility of interactions between the uncertainties, a "baseline" selection was made for each module. This was done by selecting the parameter with the

largest percent difference. For the compressors (fan, booster, HPC), this was the corrected flow. For the turbines (HPT and LPT), this was the adiabatic efficiency. Using this "baseline" setting, a series of partial derivatives of sorts was taken for the sensitivity analysis. This was done by calculating how the relative size of the uncertainty band changed by choosing the other variable relative to the baseline for all five components.

The relative size of the uncertainty band was considered for two different tool sets: a low fidelity tool set (where the uncertainty band is 100% of the value observed for actual tools) and a high fidelity tool set (where the uncertainty band is 5% of the value observed for actual tools). The two different tool sets were examined in the sensitivity analysis in order to gain insight into whether a small change in the parameter or a large change in the parameter would impact the uncertainty band size. The results from this study indicated that while the initial assumptions for the fan, booster, and turbines were correct, the uncertainty in the two thrust conditions from the uncertainty in the HPC was greater than that from the corrected flow in this component. After these findings, the new "baseline" was established. In order to ensure that this was the setting for each of the variables which would give the largest variability in the two thrust points, the partial derivative study was repeated. This study confirmed the initial results. The final results are provided in Table 18.

Table 18 is divided into five sections of rows which are separated by double horizontal lines. The top section of rows contains the headers. The five right-most columns correspond to the partial derivatives from the new baseline, which is provided in the 3^{rd} column. The second section of rows indicates which variable was being used; a 1 means that variable was used, and a 0 means it was not used. For example, the "Baseline" column has a 1 for Fan and Booster W_C and HPC, HPT, and LPT η. The fan partial derivative column switches the fan variable but leaves all others the same, and so on moving to the partial derivative columns for the other components. The third section of rows correspond to the size of the uncertainty band for the high fidelity tools (5% of Max) and low fidelity tools (100% of Max). Within each tool set, the two thrust bands are given. For example, the baseline high fidelity selection has an uncertainty band of 153.49 lb_f for maximum thrust and an uncertainty band of 30.81 lb_f for minimum thrust. The fourth section of rows then

Table 18: Sensitivity Study Final Results for Example Problem.

	Variable	Baseline Setting	Fan Partial Deriv.	Booster Partial Deriv.	HPC Partial Deriv.	HPT Partial Deriv.	LPT Partial Deriv.
Fan	W_C	1	0	1	1	1	1
	η	0	1	0	0	0	0
Booster	W_C	1	1	0	1	1	1
	η	0	0	1	0	0	0
HPC	W_C	0	0	0	1	0	0
	η	1	1	1	0	1	1
HPT	W_C	0	0	0	0	1	0
	η	1	1	1	1	0	1
LPT	W_C	0	0	0	0	0	1
	η	1	1	1	1	1	0
5% of	$F_{n@PC=50}$	153.49	83.25	151.23	152.48	153.68	153.49
Max	$F_{n@PC=26}$	30.81	17.03	30.17	30.71	30.81	30.80
100% of	$F_{n@PC=50}$	2,557.5	1,745.6	2,471.7	2,465	2,551.7	2,548.8
Max	$F_{n@PC=26}$	499.36	346.67	493.96	490.44	501.25	499.88
5%	$F_{n@PC=50}$	1.000	0.542	0.985	0.993	1.001	1.000
Ratio	$F_{n@PC=26}$	1.000	0.553	0.980	0.994	0.997	1.000
100%	$F_{n@PC=50}$	1.000	0.683	0.966	0.964	0.998	0.997
Ratio	$F_{n@PC=26}$	1.000	0.694	0.989	0.982	1.004	1.001
Ratio Mean		1.0000	0.6180	0.9801	0.9833	0.9998	0.9993

determines the ratio of the given partial derivatives thrust band relative to the baseline. If the thrust band ratio is greater than unity, it has a larger impact for that variable, and if not, it has a smaller impact. By definition, one can see that the baseline column for this section of rows is all 1.000. The final section of rows is the mean of all ratios. This was used to determine whether or not a given variable was more or less important in contributing to the uncertainty band for these two outputs.

From Table 18, one can see that the fan choice is clearly correct, and that while the others are correct, there is not a significant difference in uncertainty band size when choosing one over another. This is especially interesting when considering that the turbine uncertainty was *significantly* larger than the corrected flow uncertainty. The findings in Table 18 confirm that the new baseline settings (fan and booster W_C and HPC, HPT, and LPT η) are the outputs from these components which have the largest impact on the uncertainty band size for these two responses.

9.3.3 Example Problem Discrete Optimization Implementation

The GA implemented to solve the FSP for this example problem was very much similar to the one used in Chapter 8. The only significant difference is in the objective function. As was stated in Chapters 6 and 8, the static penalty function for the GA calculates a normalized square root of the sum of squares for all violated constraints. In the "canonical" FSP implementation in Chapter 8, this was used, but there was only a single constraint. For the example problem, there were two different thrust points being examined (net thrust for PC=50 and PC=26).

Because the uncertainty in the uncertainty – time cost trade-off was fictionalized using a single tool set, the time cost was also fictionalized. Each analysis was assumed to have a cost between 0.05 and 1, meaning that for the complete engine analysis, the time cost is between 0.25 and 5. As was the case with the "canonical" FSP discussed in Chapter 8, an uncertainty constraint was used rather than a run time constraint. Additionally, the constraint was examined from 90% to 10% of the maximum uncertainty at 20% intervals for each constraint. A table of these uncertainty constraints is provided in Table 19.

Table 19: Example Problem Uncertainty Constraint Settings.

% of Maximum Uncertainty	$F_{n@PC=50}(lb_f)$	$F_{n@PC=26}(lb_f)$
90%	2,301.75	449.42
70%	1,790.25	349.55
50%	1,278.75	249.68
30%	767.25	149.81
10%	255.75	49.94

Each of the five constraint settings provided in Table 19 were examined to test solver fitness, as was done in Chapter 8. Similarly, each point was rerun 10 times to gain some measure of variability in the final answer obtained. Additionally, the GA ran for 3,000 function evaluations (as was the case in Chapter 8) for each of the 15 scenarios considered.

9.3.4 Example Problem Uncertainty Quantification Implementation

When quantifying uncertainty in the example problem, the process outlined in Chapter 6 was implemented. This approach with FFPSO was used rather than the strategic distribution creation method discussed in Chapter 4 because there are five input variables for the analysis. The restart rule developed in Section 6.3 was implemented. Based on trends established for the three selected test functions, this should maximize the ability to capture the bounds of the space while minimizing the amount of time spent capturing them. The restart rule also stated that if after three restarts the same bounds had been consistently found (meaning there was confidence that the result was the true bounds) or a sample existed more than $1.5\bar{\sigma}$ from the current $\bar{\mu}$, then the method could terminate. This bounds convergence algorithm allowed for the uncertainty quantification portion of the experiment to be easily called in lieu of the previous simple academic function uncertainty implementation discussed in the previous chapter.

One additional development that was particularly useful in saving run time was to store previous solutions. The GA implemented in the previous chapter had a cheap function

evaluation; this is not the case for FFPSO with the convergence algorithm. While the FFPSO was the least expensive alternative, it was still not a "cheap" evaluation. For this reason, a database was developed for previous solutions to be stored.

9.4 Results

After running the GA to find the most efficient tool set, the remaining tool combinations in the full factorial were examined. In doing this, the full factorial of the cases could be considered and optimality could be assessed. For reference purposes, the full factorial of data, along with the various constraints and optimum solutions, are provided for the 15 scenarios in Figures 112 through 116. These figures are grouped such that a given tool spacing – front shape are provided in each one; in moving down the figure (top, middle, and bottom row of panes), one can see the effect of increasing the number of analysis choices. In the left column of panes for each figure, the time cost is plotted on the horizontal axis and the magnitude of the uncertainty band in $F_{n@PC=50}(lb_f)$ is plotted on the vertical axis. Similarly, the right column of panes has the same horizontal axis and plots the other uncertainty metric considered, $F_{n@PC=26}(lb_f)$, on its vertical axis. In each of the panes, the possible combinations' resulting uncertainty is provided with a black dot. The constraint at 90% of the maximum uncertainty is shown in blue, and the color approaches red as the constraint becomes more stringent. Additionally, the optimum for a given constraint is indicated with an asterisk ($*$) in the same color.

The first set of scenarios considered was for scenarios 1, 6, and 11, which have 2, 5, and 8 choices per CA, respectively. The scatterplots for these three full factorials are provided in Figure 112. In considering these panes, one can see that the front of possible choices for each uncertainty parameter looks like a series of evenly spaced ellipse-like shapes which are completely distinct from each other. The reason for this is because there is one variable dominating most of the uncertainty in both responses. This is an effect which typically happens in physical systems where one or two different components are dominating an effect. For the uncertainty in these two thrust points, there is a single component dominating: the fan. Physically, this is happening because the fan produces a significant amount of

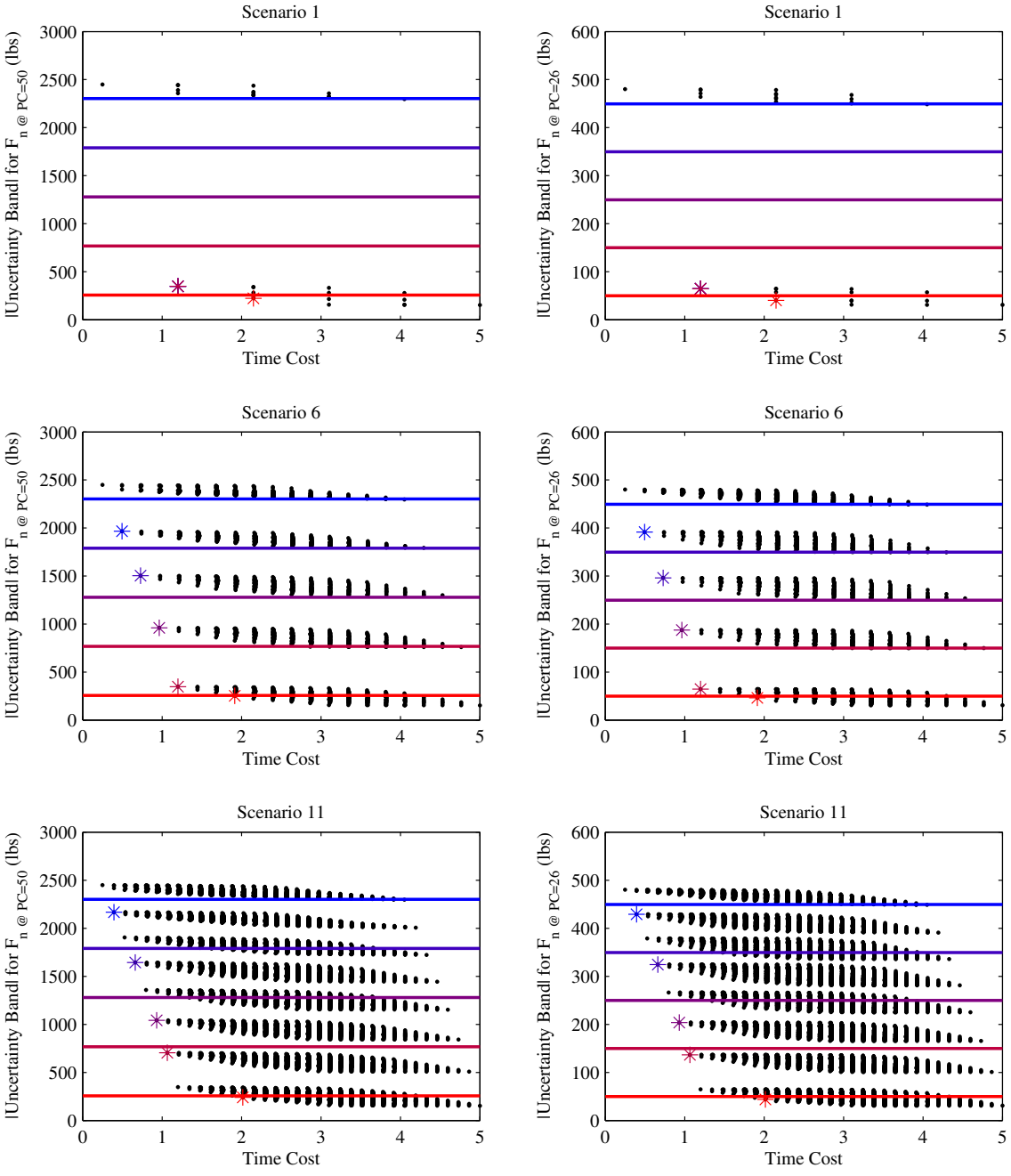


Figure 112: Scatterplot of All Possible Combinations for Scenarios 1, 6, and 11.

thrust. Interestingly, the fan had a relatively low uncertainty from a percentage perspective, especially relative to components such as the turbines, which have an uncertainty of up to approximately 26%. This means that the relatively small uncertainty in the fan is *much* more significant than the very large uncertainty in the turbines (because the fan produces most of the thrust). The reason for the even spacing is because these three scenarios all have linear front shapes and the analyses are evenly spaced.

As one examines each of these figures, one will notice that the number of ellipse-like shapes is the same as the number of choices per CA, and that as one moves from the top row of panes to the bottom row of panes, the ellipse-like shapes become more "dense".

Moving on to the second front shape – choice spacing set, scenarios 2, 7, and 12 are provided in Figure 113. This front shape – choice spacing set corresponds to an inward arching front with choices clustered on the right. This means there are a lot of low uncertainty, high cost choices, and relatively few low uncertainty or low cost choices. This can be seen, somewhat, in Figure 113. Unlike the evenly distributed ellipse-like shapes seen in Figure 112, these options have two large "clumps" of points with smaller "clumps" in each. There are still distinct groups which were seen earlier; this is due to the shape of the uncertainty – time cost trade-off. Because the components, excluding the fan, are not significantly contributing to the uncertainty in either requirement, the "clumps" one can see which move to the right (increasing time) are caused by the spacing of points along the frontier. Remember that for this front shape – choice spacing combination the choices are clumped on the right, which means there are, in effect, many choices which have a low uncertainty, but there is a large variation in the time cost associated with them. This is most clear in the middle row of panes (scenario 7). In the top row of panes (scenario 2), there are too few points to clearly see any trending, and in the bottom row of panes (scenario 12), there are so many choices on the frontier that the "clumps" one can clearly see in the Scenario 7 plots are merging with each other.

Another interesting point in this figure is that the first two constraint settings (90% and 70% of maximum uncertainty) allow for more uncertainty than any combinations in this scenario. The reason for this is because the constraints were set independently of any

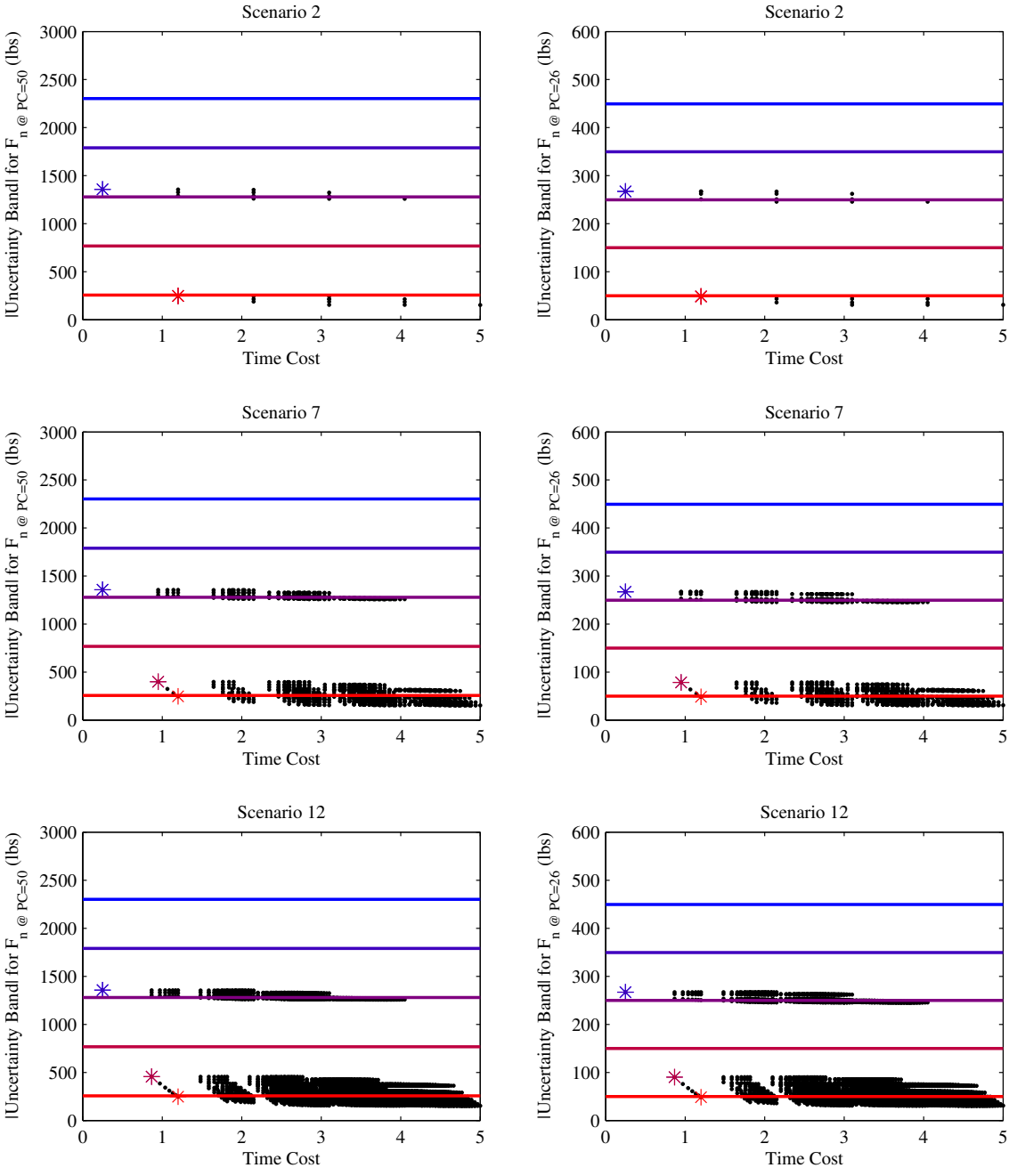


Figure 113: Scatterplot of All Possible Combinations for Scenarios 2, 7, and 12.

scenario, and for this shape front – choice spacing combination, there simply are not any extremely low uncertainty choices. Because of this, the optimums for the 90% and 70% constraint settings are the same (the asterisks are on top of each other in Figure 113). This also happens for the 50% and 30% constraint settings for several of these scenarios, but this is due to the fact that the fan drives most of the uncertainty and there is a large "space" between the cheapest choice and the next cheapest one.

Moving on to Figure 114, one can see all possible combinations for scenarios 3, 8, and 13. Scenario 3 is provided in the top row of panes, scenario 8 in the middle row, and scenario 13 on the bottom row of panes. When examining the results, the "clustering" with respect to uncertainty that was seen earlier still occurs, but interestingly, the "clustering" with respect to the time cost is significantly more clear than was the case in Figure 114. The reason for this more distinct spacing has to do with the shape of the front and the choice spacing on it. These three scenarios all have an outward arching front with the choices clustered on the right. This means that with respect to both time and uncertainty, there is a very distinct series of "clumps"; this is clearly evident in the plots for scenario 8. One should notice that in both the scenario 8 and 13 results, the top-most "clumps" of combinations are a good distance from the other "clumps". This is because the tool spacing – front shape for these scenarios has one choice for each code in the top left portion of the front and the next choice has a much lower uncertainty. As was the case in some of the other figures, the optimums for the 90% and 70% constraint settings are the same solution. The asterisks for each optimum are on top of each other, which is why there are between 2 and 4 asterisks in these figures.

Figure 115 contains a scatterplot of all possible combinations for scenarios 4, 9, and 14. The top row contains scenario 4's plots for the two thrust requirements, the middle row contains scenario 9's plots, and the bottom row contains the scenario 14 plots. The results provided in Figure 115 look very much like those in Figure 114, except reversed. The reason for this is that one could say that the front shape – choice spacing is somewhat reversed. Scenarios 3, 8, and 13 contain data for an outward arching front shape, clustered on the right, whereas this contains data for an inward arching front shape with choices spaced on

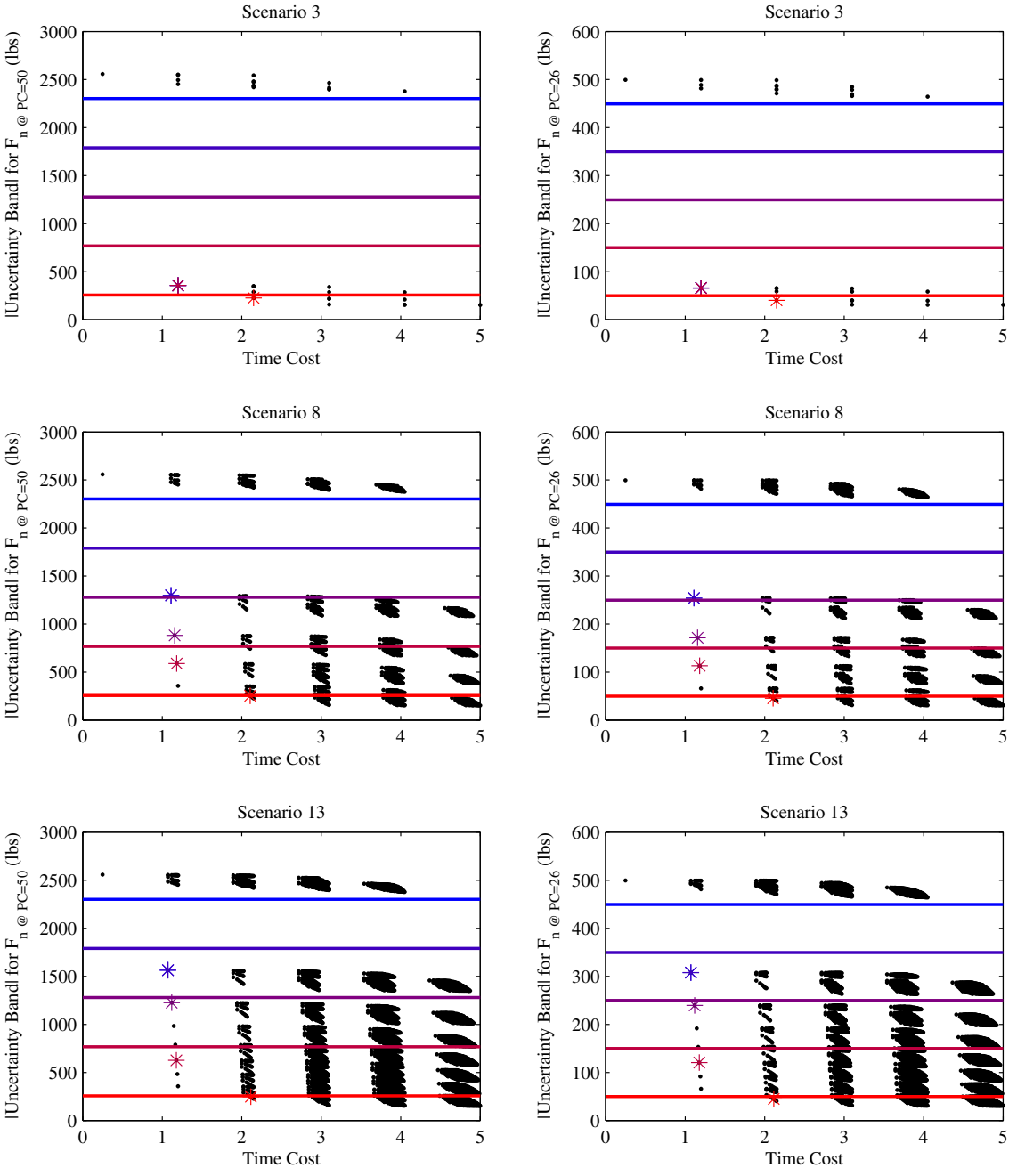


Figure 114: Scatterplot of All Possible Combinations for Scenarios 3, 8 and 13.

the left.

One should notice that while the data is very similar, there are clear distinctions as well. Figure 114 has much more distinct "clumps" of data. The reason for this is because of how the front shapes are created. While the exponents used to create them are very similar, they are not the same. While the uncertainty "clumping" is less distinct, the time cost spacing is similar. This is because the time cost spacing is assigned independently of the uncertainty front shape and is simply a reference time cost. It is interesting to notice that of all the fronts examined, this has the fewest options with a moderate to high uncertainty. The reason for this is the front shape and choice spacing, and in examining this figure, the optimum for many of the lines (indicated with an asterisk) are on top of each other.

The final front shape – choice spacing set examined was for an outward arching front shape with choices clustered on the left. The results for this are provided in Figure 116. The data for scenario 5 is provided in the top row, the data for scenario 10 is in the middle row, and the data for scenario 15 is in the bottom row. Much like the other figures, the two columns are very similar, though there are some slight differences. One should notice that the data in Figure 116 is much like the data provided in Figure 113. The reason for this is because this front shape – choice spacing set is essentially a "flipped" version of the one shown earlier. Some differences between the two figures exist, specifically that the different "clumps" of points in Figure 116 are more spaced out than those provided in Figure 113, but they have a similar shape. This is like the relationship between Figures 114 and 115.

This information was provided to give the reader an idea of the space the GA was operating upon. Regardless of the space, every scenario's uncertainty is driven primarily by the fan component. While this makes the problem less "interesting" from an optimization perspective because one parameter is driving most of the uncertainty instead of having it distributed equally over all of the parameters, it is an engineering example problem representing an actual system, not a fictitious problem with no "real world" significance. Also, by examining each of Figures 112 through 116, the reader can gain some insight into how the various scenarios are different and why the resulting optimum is different.

Strong similarity can be seen between the results for scenarios 1 through 5. The reason

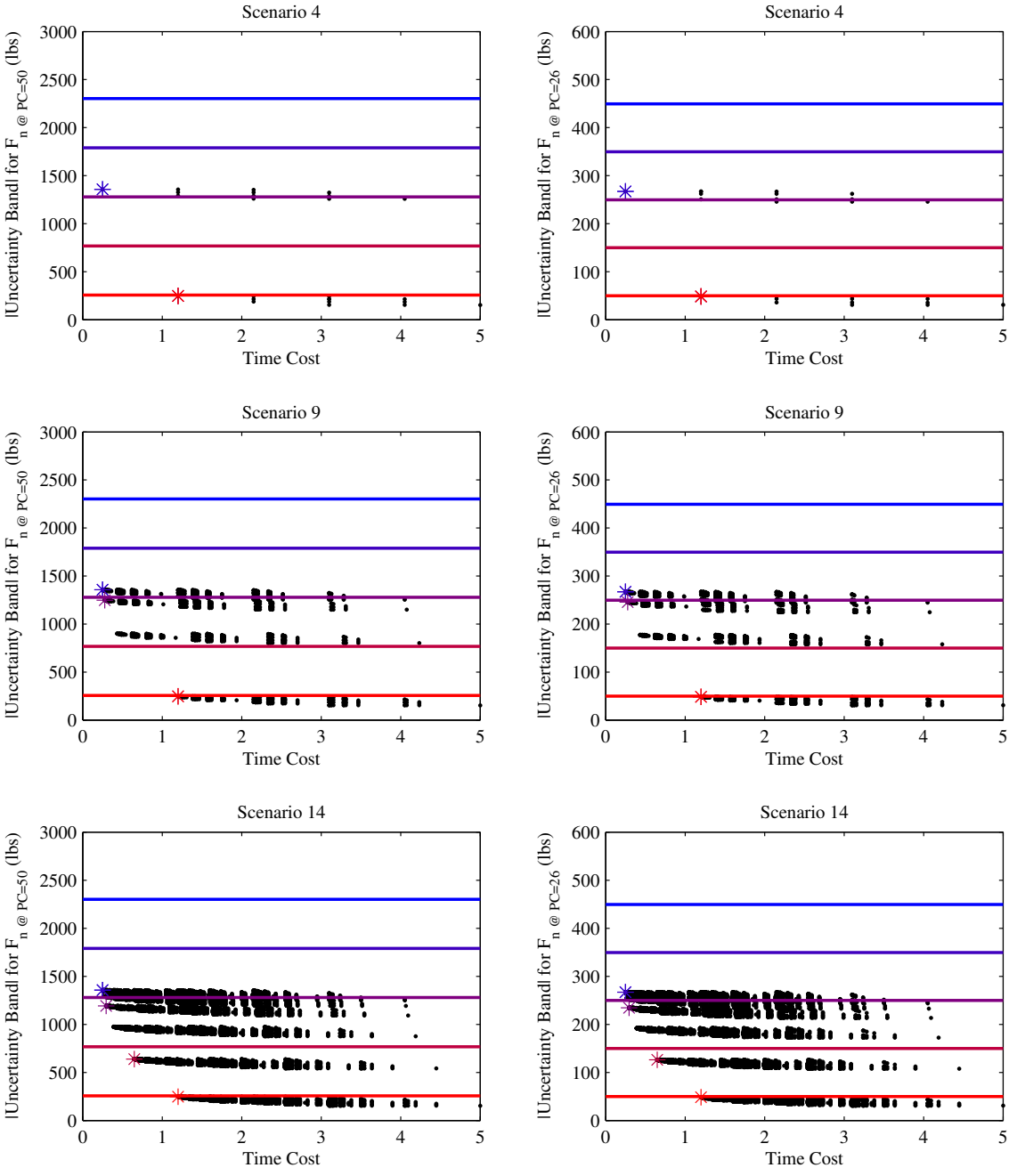


Figure 115: Scatterplot of All Possible Combinations for Scenarios 4, 9, and 14.

for this is because while each has a different front shape and spacing option, the real differences between the shape front and the spacing of analyses on it cannot really be extracted when there are only two options to choose from. These differences can most easily be seen in the data for scenarios 6 through 10. Each of these scenarios had five choices on each front. This allowed for one to clearly see the differences between the various scenarios because there were enough options to see the differences, but there were not so many options that the "clump" characteristics for that front shape – choice spacing combination were muddled together. This was the case for scenarios 11 through 15. Because of the large number of choices, these problems should theoretically be the most difficult to solve, but whether or not this is the case should be determined through experimentation.

While two metrics were identified for this illustrative example problem, they are very nearly the same. This can be seen by comparing the left and right columns in Figures 112 through 116. The intent in selecting two thrust constraints was that as the throttle setting for the SFTF decreases, the thrust contribution from the fan decreases more quickly than from the core, and the hope was that at the lower throttle setting the engine core (booster, HPC, HPT, and LPT) would become more significant. Unfortunately, this difference was not significant.

After having identified the optimum for each scenario considered, the remaining (and most important) question is whether or not the GA found it. The answer to this optimality test is displayed in Figure 117. Each pane in this figure represents a scenario, beginning with scenario 1 in the top left corner and ending with scenario 15 in the bottom right corner. Each pane's horizontal axis is the constraint setting (90%, 70%, 50%, etc) as a function of maximum uncertainty possible. The vertical axis in each pane is the percentage of iterations (for each scenario and constraint setting) where the GA found the correct answer. Recall that there were 10 iterations, so this percentage is of 10 samples. Quickly examining this figure, one can see that all panes are completely full of green bar (meaning 100% of the samples found the correct answer), with the exception of the pane for scenario 11. In this pane, only 90% of the samples for the constraint at its maximum setting (90% of the maximum uncertainty) found the correct answer. This means that of the 750 times

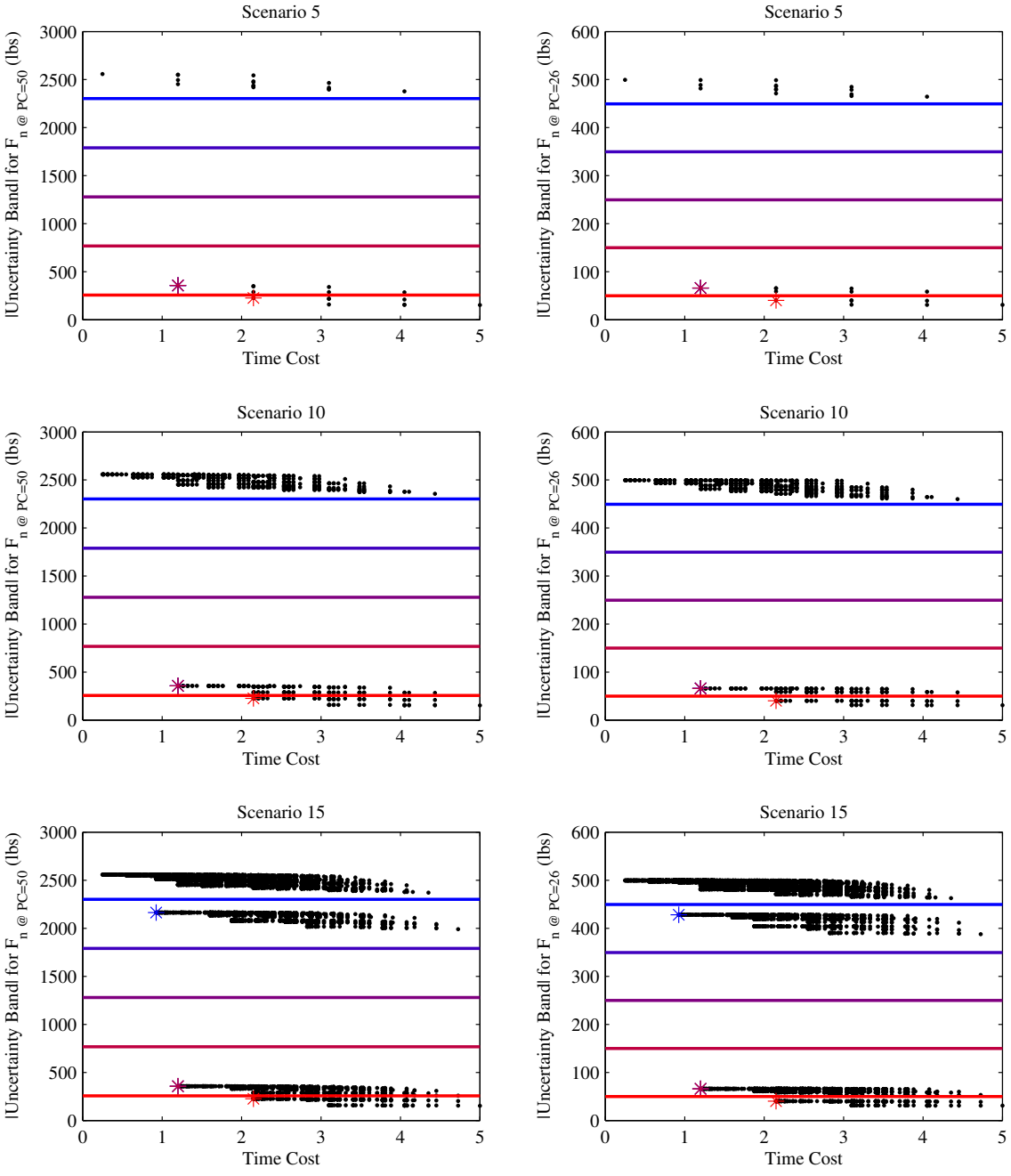


Figure 116: Scatterplot of All Possible Combinations for Scenarios 5, 10, and 15.

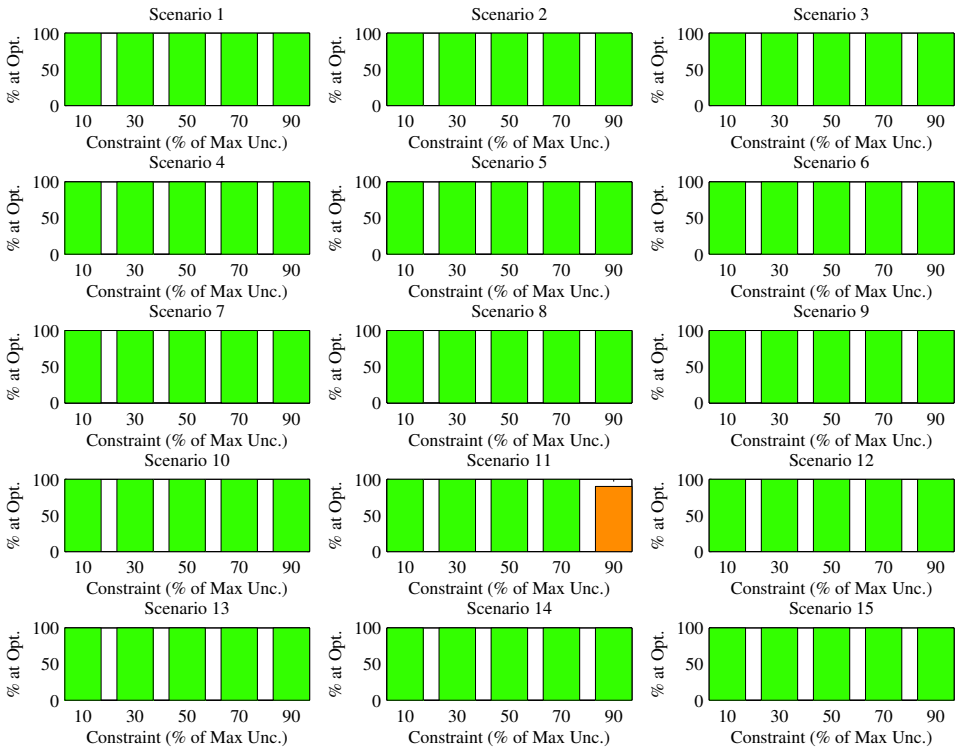


Figure 117: Histogram of Percentage of Iterations that the GA Found the Optimum by Scenario and Constraint Location.

the GA was run to create this figure, it found the correct answer 749 times (99.86% of the time).

One interesting point is that for this specific problem, the GA was allowed to run for up to $3,000$ function evaluations. This was done to maximize the probability of the GA finding the correct answer by being allowed to explore more of the space to ensure it had found the correct answer. When the GA was evaluating this problem, it would very quickly (within approximately one hundred function evaluations) find the correct answer. Therefore, if used in the future, it would likely be beneficial to develop a relative convergence criteria for the GA rather than running a pre-specified number of function evaluations. This was not developed for the example problem, because if one were to do this, they would simply be developing a rule for this example problem. The GA's ability to converge on the solution is not only a function of the number of nodes available, but also the physics of the problem which drive the FFPSO convergence on the uncertainty bounds. For this reason, rules should be developed with caution and will likely be specific to the problem they are developed for,

not some general case.

9.5 Conclusions

In this chapter, an illustrative example problem was identified to show potential users how the uncertainty propagation implemented in Chapter 6 could be implemented to quantify uncertainty for a real problem and then use those findings to determine which models should be selected using the procedure implemented for a canonical FSP in Chapter 8. An SFTF gas turbine engine was selected to illustrate the method. Uncertainty in the various analysis tools used was quantified using the E^3 test data as truth data.

Once the uncertainty in the models was quantified using the developed FFPSO, the uncertainty – time cost trade-off was fictionalized somewhat by reproducing it using adders and offsetting the epistemic uncertainty distribution. The GA developed in Chapter 8 was then used to select the cheapest set of tools, from a time cost perspective, which could provide a sufficiently small uncertainty band in the result. Two different thrust points were considered for the illustrative example, the net thrust at PC=50 and at PC=26. Several different constraint settings were examined, varying between 90% and 10% of the maximum uncertainty possible. Each of these constraint settings was considered for the 15 scenarios developed in Chapter 8.

The results from the experiments showed that the GA was able to find the correct answer for all iterations and constraint settings, with the exception of one iteration for scenario 11 with the constraint at 90% of the maximum uncertainty possible. This means the GA was able to find the correct answer for 99.86% of the times it was run for this illustrative example.

CHAPTER X

SUMMARY OF METHOD SPECIFICS

Throughout this document, the Fidelity Selection Problem (FSP) has been addressed and a method for solving it was created. Chapters 4 and 6 discuss methods for quantifying uncertainty present in analyses. Chapter 8 discusses methods for exploring the discrete optimization aspect of the FSP. Chapter 9 provides an illustrative example of how one might solve the FSP for an engineering problem. This chapter will serve to summarize this work, identify the alternatives found for each aspect, and provide specifics for the best performing options.

10.1 Uncertainty Quantification

This method is intended to be applied to a multi-fidelity Design Structure Matrix (DSM), as is illustrated in Figure 118. This example is for gas turbine engine design, though a similar system could be constructed for many other applications. As with typical DSMs, it is assumed that one will begin in the top left corner and move diagonally down the DSM towards the bottom right corner. Linkages between analyses are shown with black lines. Any lines on the top right side of the analyses represent information feeding forward. Any lines on the bottom left side of the DSM represent information feeding backward in the DSM. When this happens, one must then rerun the environment beginning at the analysis where information fed back. The depth in the chart, which is not typically shown in a DSM, represents the multi-fidelity aspect by showing various alternatives for each analysis in the DSM.

Because of the sparse data available to quantify uncertainty in the models used, an evidence theory formulation was adopted. When quantifying epistemic model uncertainty, an "adder" approach was taken in this research. This is illustrated graphically in Figure 119. In this approach, the known analyses users already possess have an "adder" distribution applied to them. The ranges for this "adder" are obtained from available, representative

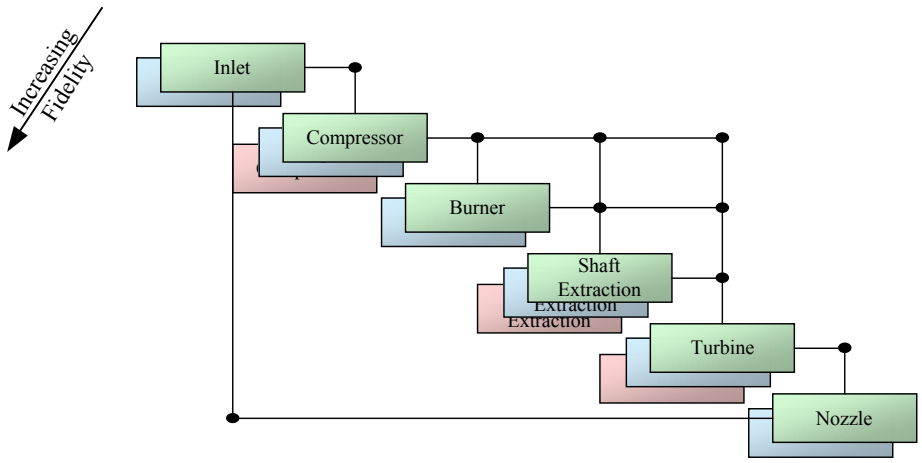


Figure 118: Example Multi-Fidelity M&S Environment.

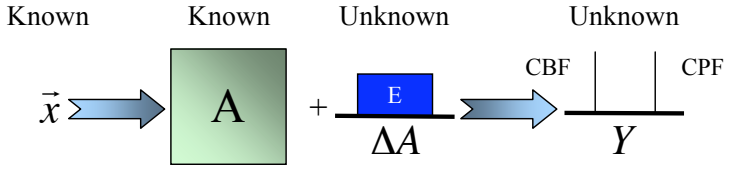


Figure 119: Proposed Approach for Modeling Epistemic Model Uncertainty in Analyses

test data. An illustration of how this might be done is provided in Chapter 9.

When quantifying the uncertainty in Y given the "adder" distributions which are applied to each analysis, a series of issues must be addressed. Given the evidence theory formulation for this study, the epistemic uncertainty distribution (the distribution with an "E" in Figure 119) to have any possible non-infinite shape. A series of experiments were conducted to most effectively represent any possible non-infinite distribution and determine how many non-infinite distributions are necessary to represent any possible non-infinite distribution.

To create any possible non-infinite distribution, a novel approach was created which strategically creates beta distributions using a space-filling Design of Experiments (DoE) to sample the beta distribution space. This approach was discussed in detail in Section 4.3.

After a number of different distribution shapes have been created, the "adder" distributions in the DSM must be sampled. The literature states that the full factorial must be considered. This would involve sampling every possible combination of distribution

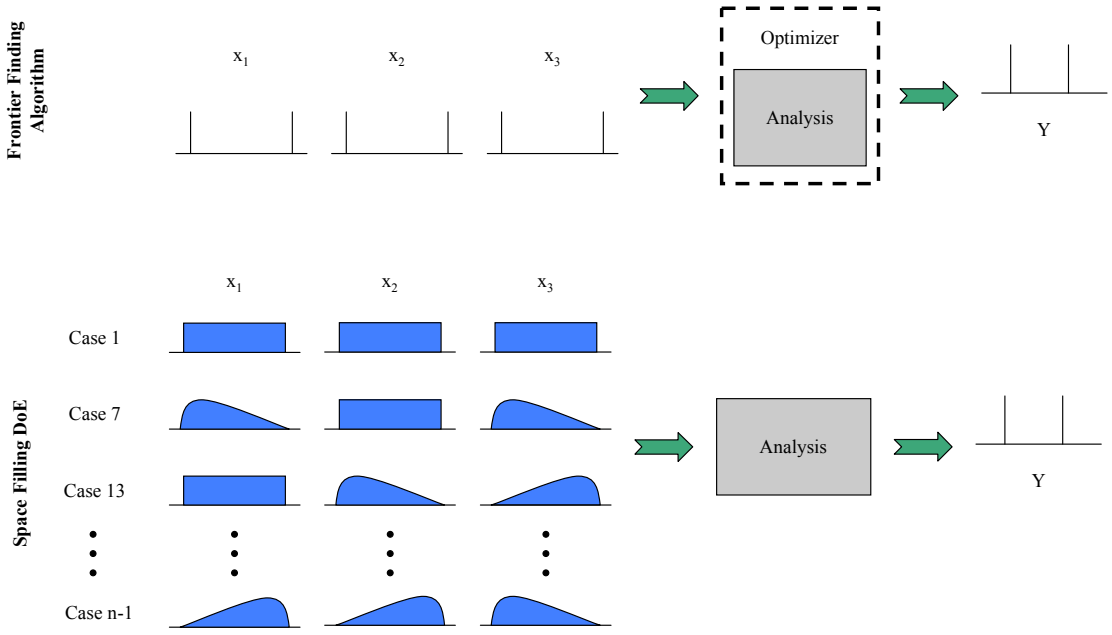


Figure 120: Illustration of Sampling Process for Two Approximations of a Full Factorial of Distribution Shapes to Capture the Bounds of Space.

shapes on the "adder" distributions in the DSM of interest in order to capture the maximum possible variability in the response(s) of interest. This was explored but found to be computationally infeasible for moderate- and high-dimensional problems. The specifics for implementing this full factorial approach are described in detail in Section 4.4.2. The dimensionality with respect to uncertainty quantification is the number of parameters where uncertainty is available for a given DSM, or equivalently, the number of "adder" distributions.

Two alternatives to the full factorial were identified. The first was to use a space-filling Design of Experiments (DoE) to strategically sample the full factorial of distribution shape combinations. This was found to significantly outperform the full factorial, but it also becomes computationally infeasible if more than four "adder" distributions exist for the DSM. The specifics for implementing this are described in Section 4.5, and it is graphically illustrated in Figure 120.

The second alternative involved using modified multi-objective metaheuristic optimizers. Rather than creating distributions to find the true variability, or bounds, of the space,

this alternative seeks the bounds directly and could be thought of as an abstraction of the full factorial approach. Three algorithms were considered when seeking the bounds of the space, a Frontier Finding Real-Coded Genetic Algorithm (FFRCGA), a Frontier Finding Continuous Ant Colony Optimization (FFCACO), and a Frontier Finding Particle Swarm Optimization (FFPSO). These algorithms were modified to work on continuous problems (they are typically formulated for discrete problems in the literature), to be multi-objective (rather than single objective), and to have a series of sub-populations, each seeking a different nondominated front. This is discussed in more detail in Chapter 5. After creating each method, the need for a restart was addressed and a relative convergence rule was created, which would terminate if:

1. Samples consistently find the same extreme bounds (the bounds have been found).
2. Samples do not consistently find the same extreme bounds (the bounds may not have been found).

If three samples each found the same extreme bounds, the first condition would terminate. If a sample was found which had a $\bar{\mu}+1.5\bar{\sigma}$ larger bound than all previous samples and at least three samples had been taken, the second condition would terminate. The 1.5 in this relationship was found to have good performance for an arbitrary representative result from the frontier finding algorithms. This process is described algorithmically in Listing 6.1 in Chapter 5.

After assessing each methods' performance, the FFPSO was found to significantly outperform both the FFRCGA and the FFCACO. A restart "rule" was also developed, though one should note this was done for a specific set of problems and, for the best possible performance, a rule should be developed for the user's specific problem class.

When comparing the space-filling DoE performance to that of the FFPSO, the space-filling DoE was found to have superior performance with restart rules 5 through 8 (discussed in detail in Section 4.6.2.1) when finding the bounds for problems with four or fewer "adder" distributions. The FFPSO was found to have superior performance for the selected test functions given the settings provided in Table 20 when finding the bounds for problems

Table 20: FFPSO Parameter Settings Found for Best Performance on Test Functions.

Parameter	Value
ω	0.95
ϕ_1	0.7
ϕ_2	0.7
Sub-population	$1,000$ per front
Max. # of Leaders	15% of sub-population

with five or more "adder" distributions. The number of fronts in the table is equal to 2^{ny}, where ny is the number of outputs of interest. For more information on how these parameters might be used, the reader is encouraged to consult Chapter 5.

10.2 Discrete Optimization

The FSP fundamentally addresses the trade off between the cost in obtaining a result and the quality of the result that is obtained. Given a Modeling and Simulation (M&S) environment, the problem is that the uncertainty in the result from a given analysis is not of interest but the uncertainty in the M&S environment output is. Unfortunately, the mapping from the analysis output to the system level output is completely dependent on the analysis being conducted and is not known a priori. The same statement could be made for the time cost in a result, as the aggregate time cost for using a given analysis is dependent on other analyses selected and any loops which must be converged for consistent physics in the M&S environment. This phenomena is illustrated in Figure 121 for a gas turbine engine. In this illustration, the time cost in a compressor aerodynamics model is being traded. While the uncertainty in the compressor outputs and the time cost for running the compressor analysis are known, this information is not relevant when solving the FSP. Instead, one is only interested in the system level uncertainty and the aggregate time cost.

Because the characteristics of this uncertainty – time cost trade off are not known a priori, one must adopt an optimization method which will consistently find the correct answer regardless of the specifics of this trade off. To accomplish this, a scenario-based

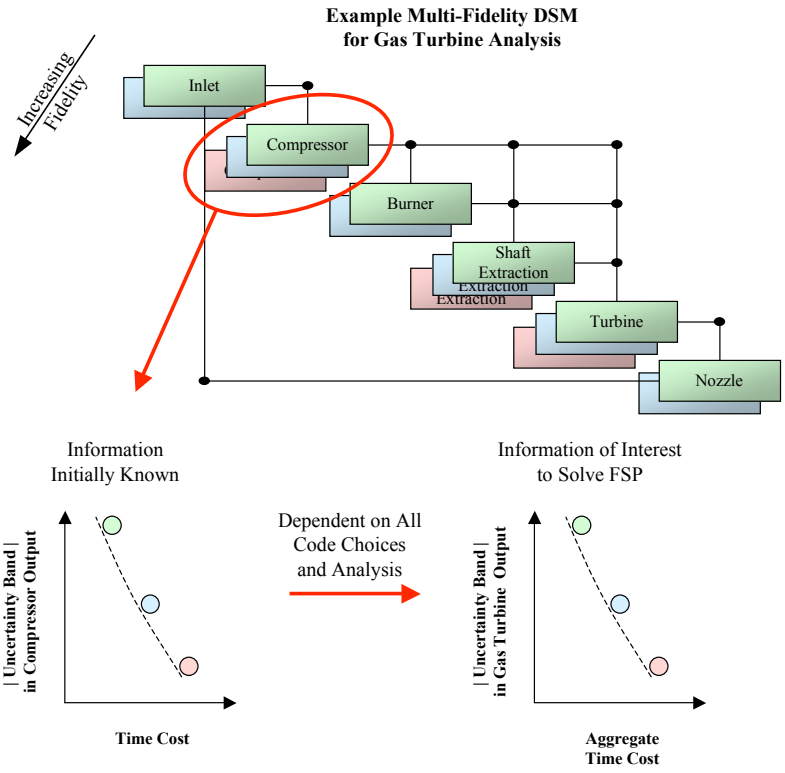


Figure 121: Notional Illustration of FSP for a Gas Turbine Engine.

approach was taken for this research. Fifteen best- and worst-case scenarios were identified and evaluated with a variety of candidate optimization methods, including a GA, an ACO, and a PSO. These algorithms were similar to those used for uncertainty quantification, but with some subtle changes. The discrete forms of these algorithms were of interest rather than the continuous form. Also, there are either uncertainty or time cost constraints when solving the FSP; therefore, a penalty function approach was adopted. This new objective function is provided in Equations 35 and 36. For more information on other alternatives which were not implemented, the reader is encouraged to consult Section 7.7.

$$h(x) = \begin{cases} f(x), & \text{if all } g_i(x) \leq 0 \\ \sqrt{\sum_{\text{active } g_i} g_i(x)^2}, & \text{if any } g_i(x) \geq 0 \end{cases} \tag{35}$$

$$g_i(x) = \left(\frac{\text{Uncertainty in } Y_i - \text{Uncertainty Constraint for } Y_i}{\text{Uncertainty Constraint for } Y_i} - 1 \right) * 100 \tag{36}$$

Where:

Table 21: GA Parameter Setting Found to Have Best Performance When Solving the FSP.

Parameter	Value
Mutation Type	Single Bit Mutation
P_{mutate}	0.5
Crossover Type	Two-Point Crossover
$P_{crossover}$	0.5
$Population$	25

- $h(x)$ is the new objective function
- $f(x)$ is the sum of aggregate run time for a given tool combination
- $g(x)$ is the percent violation of an uncertainty constraint for output i

When evaluating the 15 scenarios, the GA was found to consistently find the correct answer and to do so in a comparatively predictable number of function evaluations. The parameters for the GA found to have the best performance when solving the FSP are provided in Table 21.

10.3 Conclusions

This chapter has served to provide specifics on how one might implement this proposed method to solve the FSP for an engineering problem. This is intended to make the method more "digestible" and clarify some of the options which one might use to solve the various aspects of the proposed process. An evidence theory formulation was taken for quantifying epistemic model uncertainty and this approach employs an "adder" function to represent the uncertainty present in the analyses being used. A variety of experiments were conducted to quantify this uncertainty, and it was found that using a space-filling DoE to strategically sample a full factorial of distribution shape combinations outperforms all other explored alternatives when four or fewer "adder" functions are present. If there are five or more "adder" functions, an FFPSO was found to outperform all other alternatives. The specifics

for how these two methods might be implemented have been described in this and previous chapters.

When solving the FSP, one should choose a method capable of finding the correct answer regardless of the problem specifics, primarily because these are unknown a priori. Fifteen best- and worst-case scenarios were identified and a series of candidate optimization methods' performance was assessed for the 15 scenarios. This experiment found that a GA was able to consistently find the correct answer and do so in a relatively predictable number of function evaluations. It is therefore recommended that future users choose the GA and use it to solve the FSP while quantifying the uncertainty using either the space-filling DoE or FFPSO, depending on the number of "adder" distributions present.

CHAPTER XI

CONCLUSIONS

This chapter will begin by revisiting the research questions which led to this work as well as the hypotheses and experiments which served to answer these guiding questions. This will also include a discussion of the author's contributions to the field. Following this, the perceived use cases for the method, which were discussed earlier in Chapter 1, will be revisited. Potential future work and a second look at key assumptions which guided this research will be discussed. Finally, a summary of the research will be provided.

11.1 Revising Research Questions, Hypotheses, and Experiments

This section will revisit the research questions that guided this research, the hypotheses that were made, and experiments that were created to either test the hypotheses or answer the research questions. These research questions, hypotheses, and experiments are divided into the two focus areas for this research: uncertainty quantification and discrete optimization.

11.1.1 Uncertainty Quantification

Research Question 1: How should one create different distributions to effectively represent any possible non-infinite distribution?

This research question was answered through a literature search. A literature search was conducted and the beta distribution was found to be a good distribution for creating a wide variety of non-infinite distributions. Once the distribution type was identified, a method for strategically creating these distributions was also identified. This process involved scaling a space-filling DoE and mapping it to a and b, the beta distribution shape parameters. This was discussed in Example 1, where a method for strategically creating beta distributions is illustrated in Figure 19 and an illustration of some of the resulting beta distributions is provided in Figure 20. Both of these figures and the example are located in Chapter 4.

Research Question 2: How many different distributions are necessary to capture true

bounds of a response?

This research question acknowledges that more information can be gained by creating additional distributions of different shapes, but it is not documented in the literature how many are necessary. The answer to this research question was hypothesized. The results from Experiment 1 served to both answer this research question and test Hypothesis 1.

Hypothesis 1: The number of distributions required to capture the maximum possible variability in a response will asymptotically occur and will be dependent on the specific problem and the number of inputs being examined.

An illustration of what was expected with this hypothesis was provided at the beginning of Chapter 4. This hypothesis was tested by conducting Experiment 1. As stated in the hypothesis, it was anticipated that creating additional distributions would increase the bounds that were captured, but at some point, no new information could be gained by creating additional distributions, meaning that at that point the actual bounds had been found and an asymptote had been reached. By answering this question, one could determine whether or not value is added by creating more distributions and how many distributions would need to be created to capture the bounds of the space.

Experiment 1 was created to test Hypothesis 1 and answer Research Question 2. This experiment identified three n-dimensional optimization test functions: the Schwefel, Michalewicz, and Sphere Functions. A variety of different distributions, including a uniform distribution, was created and the bounds of the space which had been found were determined. The results from this experiment show that for the Schewfel and Michalewicz test functions a larger bound for the space was found by creating a variety of different distributions than was found using a maximum entropy approach (uniform distribution). The Sphere function was the sort of "special case" where the bounds of the problem were found with a maximum entropy approach for low-dimensional versions. What these results showed was that for the general case (and especially for more complex or moderate to high dimensional problems), a *significantly* larger bound could be found by creating additional distributions. These results confirmed Hypothesis 1, answered Research Question 2, and are provided in Chapter 4.

This did, however, lead to one problem. The literature states that because all inputs for the problem should be assumed independent, each distribution shape should be sampled with every other distribution shape. This means that a full factorial of distribution shapes must be sampled. While the results did asymptote at some point as was expected, the necessity of the full factorial mean that sampling for moderate to high dimensional problems (4 or more inputs) was computationally infeasible. This significant issue led to Research Question 3.

Research Question 3: How can one best approximate the boundary of a response obtained with a full factorial of distribution shapes when quantifying uncertainty to solve the Fidelity Selection Problem?

This research question was asked to try to create a way to feasibly find the true bounds of the space obtained in Experiment 1. In conducting a literature review for Research Question 3, two alternatives were identified:

1. Using a space-filling DoE to sample the full factorial of distributions
2. Using a metaheuristic frontier finding algorithm to find the boundary of the space

These two options each possess pros and cons, which are discussed in detail in Chapter 4. These pros and cons led to Hypothesis 2.

Hypothesis 2: A Space-filling DoE will outperform a metaheuristic frontier finding algorithm when approximating the results obtained in Experiment 1.

Experiment 2 was created to test Hypothesis 2 and provide some data for answering Research Question 3. This experiment used the full factorial data for the three optimization test functions, created in Experiment 1, as truth data. This data was only available for 2 to 4 dimensional problems for each of the three test functions, so the experiment was broken into two parts.

Experiment 2.1 was conducted to create relative convergence rules for the strategic distribution creation method as a function of the number of input variables. A variety of different rules were tested and two were found which consistently worked for the test

problems as they increased in dimensionality. The resulting rules from this experiment can be found in Chapter 4.

Experiment 2.2 was conducted to use the relative convergence rules created in Experiment 2.1 and apply them to higher dimensional forms of the optimization test problems where the actual bounds could not be feasibly obtained. The results from this experiment showed that the rules did not consistently converge on a solution in a reasonable amount of time (less than 5 hours) and there was very little confidence that the solution they did obtain was correct. The results showing poor convergence properties for these relative convergence rules can be found at the end of Chapter 4.

Research Question 3.1: What metaheuristic frontier finding algorithms will most quickly find the bounds of the space?

This research question was asked to identify which metaheuristic frontier finding algorithms could most effectively approximate the full factorial results. Because there has been a lot of significant development in the past 15 years in the area and there are a wide variety of algorithms to choose from, a literature review was in order. The findings from this literature review can be found in Chapter 5. This literature review identified three promising algorithms: GA, ACO, and PSO. These three algorithms were modified to solve continuous problems and search for the frontiers of the space, producing three new algorithms: Frontier Finding Real-Coded Genetic Algorithm (FFRCGA), Frontier Finding Continuous Ant Colony Optimization (FFCACO), and Frontier Finding Particle Swarm Optimization (FFPSO). In order to determine which of these three algorithms performed the best, Experiment 3 was conducted.

Experiment 3 used the newly developed FFRCGA, FFCACO, and FFPSO to find the boundary for the three optimization test functions. This experiment took into account the need for restarting the algorithm; the results found that the FFPSO was the most favorable algorithm. While some of this can be attributed to implementation, much of this is due the algorithm's ability to lend itself well to vectorization and the fundamental processes in PSO. PSO can be completely vectorized, unlike GA and ACO, which fundamentally requires some row by row (or equivalently population member by population member) operations.

Additionally, if one assumes that the function evaluation is inexpensive, then the most expensive operation in each of these three algorithms is the nondominated sorting which takes place to find the frontiers. In the FFRCGA and FFCACO, this involves identifying the best half of the population. In PSO, the algorithm is only fundamentally interested in the leader(s) (the first Pareto front), not the best half of the population. For this reason, less time is spent on nondominated sorting in a given generation/iteration; it is therefore a much better choice for continuous problems.

The results from Experiments 2 and 3 were compared and a "hybrid" answer to Research Question 3 was found. If there are 4 or fewer input variables for a given problem, then the strategic distribution generation method is able to most effectively find the boundary of the space. If there are more than 4 input variables, then the FFPSO is able to most effectively find the boundary of the space. These findings disprove Hypothesis 2, because it stated that the space-filling DoE approach would always be fastest; it is only fastest for low dimensional problems. The results from Experiment 3 and the data which led to this answer to Research Question 3 can be found in Chapter 6.

11.1.2 Discrete Optimization

Research Question 4: What optimization method finds the correct answer to the FSP?

This research question asks how one should go about solving the FSP. This question is the "real question" behind this research. In order to initially answer this question a literature review was conducted. This identified many possible discrete optimization methods which could be used and discussed why many of them were not applicable to the FSP. Some of the most limiting issues were that the uncertainty in a given analysis is unknown until it has been paired with all other analyses, meaning methods such as A* and D* must be eliminated, and that it is a nonlinear problem, meaning methods such as dynamic programming must be eliminated. After a qualitative comparison, the five most favorable methods were found to be GA, ACO, Artificial Bee Colony (ABC), PSO, and Firefly Algorithm (FA). Because of clear similarity between ABC and ACO as well as the clear similarity between FA and PSO, ABC and FA were eliminated from comparison and the performance of ACO

and PSO should be thought of as a first-order result from these algorithms. In order to determine which of these three algorithms could consistently find the optimum for the FSP, Experiment 4 was conducted.

Experiment 4 tests how GA, ACO, and discrete PSO perform when solving the FSP. Because much of why a user might choose one code over another for a given analysis is driven by the physics of the problem (as was shown in the illustrative example problem), using an engineering problem to determine the real fitness of a given candidate method would be questionable. Another issue for decision makers is determining how the uncertainty in an output from one analysis impacts a system level output of interest. This mapping is due to the physics of the problem and may also be dependent upon the possible uncertainty from other analyses. For this reason, effort was taken to decouple the application (physics of the problem) from the selection method. This was done by using a simple academic function in lieu of an engineering analysis. To show how the characteristics of the FSP might challenge candidate selection methods, a canonical form of the FSP was created and 15 best- and worst-case scenarios were evaluated. These scenarios were also evaluated to establish solver fitness with respect to constraint setting and problem dimensionality. These results showed that the GA selected nearly always found the correct answer. While the GA was identified as the "smart choice", the discrete PSO was able to find an answer more quickly (though it was not necessarily the right one). The experiment concluded with recommending the GA for further use when solving the FSP.

11.1.3 Summary of Contributions

The previous section discussed the research questions which served to guide this research. There was also discussion of how these questions were answered, whether through literature reviews and examples or through hypotheses and a series of experiments. While some may say that the hypotheses conducted are an author's contribution to the field, this is not necessarily the case because these hypotheses were based on available literature reviews and were intended to "guess" the answer to the research questions. One could likely consider the experiments, their results, and their overall findings to instead be the contribution. These

specific contributions are provided more explicitly for the reader's benefit.

- The author showed that the limiting case for epistemic uncertainty quantification (an interval) can be approximated using a space-filling DoE which strategically samples the full factorial of distribution shapes discussed in the literature.
- The author showed that the limiting case for epistemic uncertainty quantification (an interval) can be approximated using a variety of metaheuristic frontier finding optimizers, and that the FFPSO developed for this research consistently approximated the full factorial of distribution shapes most quickly.
- The author created a canonical form of the FSP, from which any actual set of analyses can be mapped.
- The author showed that a GA can be effectively used to find the correct answer to the FSP regardless of the specifics of the problem.
- The author showed that while a GA can be used to find the correct answer, a discrete form of PSO may find the correct answer more quickly under certain circumstances.
- The author illustrated how the uncertainty in analyses can be quantified and the FSP can be solved for an SFTF gas turbine engine example.

11.1.4 Revisiting Perceived Use Cases

In the motivation for this research, two separate use cases were outlined. The first was to use this method to determine the difference between two concepts in a down selection; in other words, use this method to pose the constraints for the FSP to provide sufficient concept resolution. The second use case was to allow users to place a series of different uncertainty constraints on the outputs from a DSM.

The first use case should be thought of as a method for justifying why one concept should be selected over another. For the purposes of this use case, this could be thought of as selecting Concept A or B. This would be formulated in much the same way as the simple academic function FSP in Chapter 8 and the illustrative example provided in Chapter 9,

but with a few slight changes. In these problems, the objective was to minimize run time and meet uncertainty constraint(s). To provide concept resolution, one should use a slightly modified objective function and constraint, as provided in Equations 37 and 38.

$$h(x) = \begin{cases} f(x), & \text{if all } g_i(x) \leq 0 \\ \sqrt{\sum_{\text{indistinct } g_i} g_i(x)^2}, & \text{if any } g_i(x) \geq 0 \end{cases} \tag{37}$$

$$g_i(x) = \frac{\text{Uncertainty Band Overlap for } Y_i}{\text{Concept A Uncertainty Band for } Y_i + \text{Concept B Uncertainty Band for } Y_i} \tag{38}$$

Where:

- $h(x)$ is the new objective function
- $f(x)$ is the sum of aggregate run times for a given tool combination ($Time_{ConceptA} + Time_{ConceptB}$)
- $g_i(x)$ is the normalized overlap of uncertainty bands for output i
- Indistinct g_i are all ouputs where the uncertainty bands from Concepts A and B overlap

Additionally, preference is always given to feasible solutions over infeasible ones. By using this objective function and constraint, the GA is used to simultaneously select the tools for the DSM for both concepts A and B.

The second use case, which was discussed earlier, was where a variety of uncertainty constraints could be placed on a variety of different outputs and one would minimize the amount of time spent conducting the analysis. In this case, one would use the objective function and constraint initially described for the FSP in Chapter 8. This formulation allows for multiple uncertainty constraints. The values could be outputs from any of the various analyses. The illustrative example and the simple academic function FSP could be thought of as part of the second use case. Up until this point in the analysis, the second use case has been utilized when one wishes to minimize run time and meet some uncertainty constraint. At the beginning of Chapter 7, the idea that the objective function and the

constraints could be switched was discussed. By this, it is meant that instead of minimizing the time cost given some uncertainty constraint, one could minimize the uncertainty given some time cost constraint. This could be thought of as the "opposite" of the second use case and would only require a trivial modification to achieve this new objective. The real question is what the user really needs: to minimize the amount of time spent but get an answer that meets some accuracy threshold or to get an answer as good as possible and not spend more than a certain amount of time getting it.

11.2 Future Work and Revisiting Key Assumptions

This section will begin by revisiting key assumptions which were made over the course of this research and then reflecting on how, if the assumptions were made differently, significantly different answers could have been obtained. There will also be some discussion on new developments in the optimization community and some discussion on convergence.

11.2.1 Revisiting Uncertainty Assumptions

One of the first assumptions made for this research was that the "reality" one could use to compare the various analysis tools against would be the thing closest to reality that is available: physical test data. A common alternative assumption is to instead treat the highest fidelity tool available as "reality". Each choice has its merits and, depending on the circumstance, perhaps a different choice should have been made. Much of this research was motivated by work on combined cycle power plant design. When considering these systems, physical test data is readily available. Various sensors can be integrated into the systems or placed at exits to take readings; these readings can then be used as "reality", as this is how the systems may actually operate.

Conversely, if one were designing a system such as a missile, spacecraft, or nuclear reactor, very little testing would likely be conducted. Placing sensors on these systems may be extremely difficult, if not impossible, and there may not be any physical test data available. If this were the case, then taking the other choice, that the highest fidelity tool available is reality, would likely make the most sense. This approach is different because instead of making assumptions with test data in an evidence theory formulation, the user must

instead assume that the highest fidelity analysis is reality. The validity of this assumption, and whether or not users are willing to make this, is left to them in the future.

If physical test data was not available and the highest fidelity analysis was chosen to represent "reality", then there would not be a sparse data problem. Data availability would only be limited by the amount of computing time available for a project. If this were the case, then a probabilistic approach to uncertainty may make more sense and much of the work conducted for this research would not be applicable. An evidence theory approach would not need to be adopted, and one could simply take a probabilistic approach as has been done for the last 100+ years. This is the approach taken by many members in the reliability field, especially when there is no physical test data available for a study.

A second alternative, which has been discussed by some, is to use a probabilistic approach to quantifying uncertainty, even given sparse data, and accept the central limit theorem's influence on the resulting uncertainty. While this may be suitable if the phenomena is random or even quasi-random, the author believes this is not applicable to this study because model uncertainty is some underlying trend or correlation which has been simplified or removed from the analysis and representing this with a normal distribution is not sufficient.

11.2.1.1 Another Perspective on Parameter and Model Uncertainty

In this study, the focus was on model uncertainty. This was roughly defined earlier to be how well an analysis represents reality. As was discussed in Chapters 2 and 8, there is a fundamental trade-off between model uncertainty and time cost. One point excluded from this discussion that was kept outside of this research's "control volume" was parameter uncertainty. In higher fidelity models, there is less model uncertainty. The reason for this is because this uncertainty is being transferred to the inputs, or parameters, for the model.

To obtain trends such as those shown in Chapter 8 for parameter uncertainty, one would need full knowledge of the parameters for the study. In Chapter 8 these were represented with the coefficients for the polynomial approximation of a turbojet powerhook. In Chapter 2, this was the temperature and pressure of the high temperature gas. If this information

was not available, then one would likely have the same (or possibly more) uncertainty by using a higher fidelity model if these parameters were unknown. This is a trade-off which is discussed in the literature [168] and is dependent on the problem. To obtain a more "realistic" measurement of the total uncertainty, one would have to examine both parameter and model uncertainty simultaneously. While this was not done in this study, the evidence theory approach which was chosen makes that choice possible for future users.

11.2.2 Promising Hybrid Methods for Optimization and Additional Methods

One area of development in recent years is the use of hybrid metaheuristic optimization methods. These methods take one of two possible approaches. The first approach is to use some sort of discrete metaheuristic algorithm such as ACO, and after each ant has finished moving through the graph, a gradient or quasi-gradient based method would be used to find a local optimum. Example of gradient and quasi-gradient methods used in this approach include Tabu Search (TS) [60] or Harmony Search (HS) [228]. The second approach of hybridization considered is to take aspects of algorithms such as crossover or mutation from a GA and implement them in some other method such as PSO or FA. This has been shown by many to be a promising area of research.

Neither of these methods were significantly implemented in this research. The first was dismissed because, while using a continuous, gradient-based optimizer at the conclusion of a discrete metaheuristic optimizer may produce favorable results, it was assumed by the author that it would be more efficient to instead convert the discrete metaheuristic optimizer to a continuous one. This was done for both GA to form Real-Coded Genetic Algorithm (RCGA) and for ACO to form Continuous Ant Colony Optimization (CACO). Embedding the continuous optimization in the algorithm itself also resolves issues with "hamming cliffs" which are a well documented problem for discrete optimizers solving continuous problems and were discussed in Chapter 5.

The second method, putting popular aspects of more developed metaheuristic optimizers in newer metaheuristic optimizers, was only explored in a limited fashion. Aspects such as putting a GA's mutation or crossover in methods such as PSO have been discussed and

implemented by some. The mutation is commonly referred to as "turbulence". This is where some random scalar is applied to the new velocity vector. This allows the "angle" the point mass is taking to change slightly and may lead to a new, better result. This was adopted in the FFPSO used for uncertainty quantification and the discrete PSO used for solving the FSP. Other more significant methods involve performing some crossover of non-leaders in PSO. While these schemes may work well for some problems, they were not explored for this one. The FFPSO outperformed both the FFRCGA and FFCACO primarily because there was less nondominated sorting being conducted but also because the process could be completely vectorized. If a crossover-like process were to be added, PSO would no longer be completely vectorized and would suffer a performance loss because of this fact. This is an area of potential future work, but the result would likely be dependent on the problem being examined and how "poor" the candidate solutions were.

Given FFPSO performance when quantifying uncertainty, the natural "next question" is how well PSO derivative methods would perform. One such method, FA [227] was identified in Chapter 5. This method was shown to be very promising and may outperform PSO given Yang's modifications. This author strongly encourages future users to explore this algorithm as a possible alternative.

11.2.3 Re-examining Uncertainty Convergence

The uncertainty convergence rule developed in Chapter 6 was done using data for the Schewfel and Michalewicz functions. This rule was developed to give future users some idea of how long the FFPSO should run before restarting. This rule, given that it is a time, not a number of generations, is dependent on the population size selected, the speed and memory of the computer it is being run on, and whether or not any other processes are being run on that machine. For this reason, it would likely be useful for future users to use this as a guideline and develop their own restart criteria. If the function being evaluated is simpler than either the Schwefel or Michlewicz functions, then it is likely that time can be saved by coming up with some shorter restart time. This was the case for the illustrative example. The restart time was *significantly* decreased because of the simplicity in an SFTF

design relative to either the Schewfel or Michalewicz functions. It is suggested that future users develop rules for their problems. This, unfortunately, is the case with all optimization problems where "rules" are applied. These serve more as guidelines and can be tweaked for the problem at hand.

11.2.4 Re-examining Fidelity Selection Problem Convergence

The research question with regard to discrete optimization asked which optimization method was able to find the correct answer. This is a very important question (and is in the author's opinion the most important question), but another important question should also be asked: how quickly can one expect to achieve this answer? For the fourth experiment conducted in Chapter 8, this was a set number of generations. For the simple academic function FSP, this question was not addressed, but as was shown, two of the three methods considered were not able to consistently find the correct answer even with restarts, so it should not impact which method should be chosen.

When the GA was used for the illustrative example problem, this was taken to be roughly $3,000$ function evaluations (or some limit on the number of generations, whichever was achieved first). Given that the fidelity in the results of interest for the example problem discussed in Chapter 9 was nearly completely driven by the fan, the GA was able to find the correct answer very quickly by roughly the 5^{th} or 6^{th} generation, and all subsequent generations served only to more thoroughly sample the points around the optimum. For this reason, it is likely that in the future, one could terminate the GA much sooner than was done for the illustrative example problem. Again, as was the case with the uncertainty convergence, this would be extremely dependent on the function of the problem at hand, it is something that should be investigated further before one attempts to solve the FSP for one's problem.

Another item which should be examined is the selection method. For the simple academic function FSP examined in Chapter 8, the shape and spacing of code choices on the frontier were varied, examining the "extremes" of the front shape – choice spacing option space. In nearly all combinations considered, which were all corners, face centered points,

and a center point, the GA found the correct answer. In some parts of the space, however, PSO was able to find the correct answer more quickly than the GA. For this reason, if one knows a priori that their analyses are in the region of the space where PSO found the correct answer, one could obtain better results with PSO.

11.3 Summary

This document discusses research conducted to solve the Fidelity Selection Problem (FSP). By solving the FSP, the user is able to determine how accurate the various analyses in their Modeling and Simulation (M&S) environment should be in order to either achieve sufficiently accurate results (for concept resolution) or to meet time constraints. Much of the current literature focuses on developing higher fidelity analyses or better analyses for a given problem. This research instead provides the user with a way to determine the situations where these new higher fidelity analyses should be used and whether or not existing choices can instead fulfill anticipated needs. In making this determination, there are two main portions: quantifying the uncertainty in the available analyses and performing the actual analysis selection.

This first portion of this research outlines how one could quantify the uncertainty already present in analyses. This research treats physical test data as reality for the purposes of uncertainty quantification. This assumption therefore means that only a limited amount of data is available. Given this data sparsity, the traditional probabilistic approach to uncertainty quantification is challenged because it intrinsically under-predicts the possible bounds of uncertainty due to the central limit theorem. Instead, an evidence theory approach is taken where the uncertainty bounds, not the probability of an event occurring, are of interest.

Because the literature is curiously lacking an engineering implementation of uncertainty quantification for a problem such as this, a novel approach was explored which assumes each input is independent of all others and which uses a full factorial of distribution shapes to quantify uncertainty. This research determined the number of different distributions necessary to find the true bounds. While this approach was extremely thorough and very

closely resembles Dempster's original discussion on the method, it is computationally infeasible for moderate and high dimensional problems. For this reason, two approximation methods were considered: using a space-filling Design of Experiments (DoE) to strategically sample the full factorial approach and using metaheuristic frontier finding algorithms to find the bounds of the space. A variety of experiments was conducted and it was found that the space-filling DoE approach was fastest when 4 or fewer inputs exist for the problem. A specific metaheuristic frontier finding algorithm, Frontier Finding Particle Swarm Optimization (FFPSO), was fastest when there are more than 4 inputs.

When performing the actual model selection to solve the FSP, a case was made to pose the FSP as an optimization problem. Relevant optimization literature was examined and most methods available were eliminated because the FSP uncertainty is likely nonlinear and because the uncertainty from a given analysis choice cannot be determined until all other analyses have been selected. For this reason, three promising metaheuristic methods were identified: Genetic Algorithm (GA), Ant Colony Optimization (ACO), and discrete Particle Swarm Optimization (PSO).

In an effort to decouple the selection process from the physics of the problem, so that the various candidate methods could be better evaluated, a canonical form of the FSP was created and a simple academic function was evaluated. Given this canonical form, any actual set of analyses, whether a gas turbine, aircraft, power plant, or any other analysis, could be mapped to the canonical form. Because of the well documented poor performance of Monte Carlo (MC) methods for sampling the entire space, a scenario-based approach was taken. The three candidate methods were evaluated for each of the selected scenarios as they scaled up with number of choices per Contributing Analysis (CA) and number of CAs in the Design Structure Matrix (DSM). The GA was found to consistently find the correct answer, though under some circumstances the discrete PSO was able to find the correct answer more quickly. For this reason, the GA was recommended for future use when solving the FSP.

In order to better illustrate how this process for solving the FSP might be solved for an engineering problem, an illustrative example problem was also created. This problem

determined what fidelity level was necessary for conducting Separate Flow Turbofan (SFTF) gas turbine engine analysis, given a variety of different constraints.

REFERENCES

[1] "Aircraft propulsion system performance station designation and nomenclature," Tech. Rep. Aerospace Recommended Practice (ARP) 755 B1, Society of Automotive Engineers, 1994.

[2] Aerospace Systems Design Laboratory, "University Research, Engineering, and Technology Institute (URETI) on Aeropropulsion and Power Technology (UAPT) Task 2.1.1 Formulation, Development and Creation of VISSTA/VIPER-CAT Environment Final Report," 2008.

[3] Agarwal, H., Renaud, J. E., Preston, E. L., and Padmanabhan, D., "Uncertainty quantification using evidence theory in multidisciplinary design optimization," *Reliability Engineering and System Safety*, vol. 85, no. 1-3, pp. 281–294, 2004.

[4] Ahuja, R. K., Hochbaum, D. S., and Orlin, J. B., "A cut-based algorithm for the nonlinear dual of the minimum cost network flow problem," *Algorithmica*, vol. 39, no. 3, pp. 189–208, 2004.

[5] Alvin, K., Oberkampf, W., Diegert, K., and Rutherford, B., "Uncertainty quantification in computational structural dynamics: A new paradigm for model validation," in *Proceedings of the International Modal Analysis Conference - IMAC*, vol. 2, pp. 1191 – 1197, 1998.

[6] Anderson, J. D., *Hypersonic and High Temperature Gas Dynamics*. American Institute of Aeronautics and Astronautics, 2000.

[7] Anderson, J. D., *Fundamentals of Aeronautics*. McGraw-Hill Publishing Company, 3^{rd} ed., 2001.

[8] Aneja, Y., Aggarwal, V., and Nair, K., "Shortest chain subject to side constraints," *Networks*, vol. 13, pp. 295–302, 1983.

[9] Ang, A. H.-S. and Tang, W. H., *Probability Concepts in Engineering: Emphasis on Applications to Civil and Environmental Engineering*. John Wiley and Sons, Inc, 2^{nd} ed., 2007.

[10] Angantyr, A., Andersson, J., and Aidanpaa, J.-O., "Constrained optimization based on a multiobjective evolutionary algorithm," in *The 2003 Congress on Evolutionary Computation*, vol. 3, pp. 1560–1567, 2003.

[11] Angus, D., *Progress in Artificial Life*, vol. 4828 of *Lecture Notes in Computer Science*, ch. Population-Based Ant Colony Optimization for Multi-Objective Function Optimisation, pp. 232–244. Springer Berlin and Heidelberg, 2007.

[12] Armand, P. and Benoist, J., "A local convergence property of primal-dual methods for nonlinear programming," *Mathematical Programming*, vol. 115, no. 2, pp. 199–222, 2007.

[13] Aughenbaugh, J. M., *Managing Uncertainty in Engienering Design Using Imprecise Probabilities and Principles of Information Economics.* PhD thesis, Georgia Institute of Technology, 2006.

[14] Bartz-Beielstein, T., Limbourg, P., Parsopoulos, K. E., Vrahatis, M. N., Mehnen, J., and Schmitt, K., "Particle swarm optimizers for pareto optimization with enhanced archiving techniques," in *Congress on Evolutionary Computation*, vol. 3, pp. 1780–1787, 2003.

[15] Bauer, J., Magee, J., Moses, G., Leitch, R., and Dawson, S., "Medical Simulation Training Initiative (MSTI)," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4037, pp. 254 – 63, 2000.

[16] Beamish, D., Scott MacKenzie, I., and Wu, J., "Speed-accuracy trade-off in planned arm movements with delayed feedback," *Neural Networks*, vol. 19, no. 5, pp. 582 – 599, 2006.

[17] Beasley, J. and Christofides, N., "An algorithm for the resource constrained shortest path problem," *Networks*, vol. 19, no. 4, pp. 379–394, 1989.

[18] Bellman, R., *Dynamic Programming.* Princeton University Press, 1957.

[19] Bellman, R., "On a routing problem," *Quarterly of Applied Mathematics*, vol. XVI, no. 1, pp. 87–89, 1958.

[20] Beltrame, G., Sciuto, D., and Silvano, C., "Multi-accuracy power and performance transaction-level modeling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 10, pp. 1830 – 1842, 2007.

[21] Bertsekas, D. P., *Nonlinear Programming.* Athena Scientific, 2^{nd} ed., 1999.

[22] Biltgen, P. T., *A Methodology for Capability-Based Technology Evaluation for Systems-of-Systems.* PhD thesis, Georgia Institute of Technology, 2007.

[23] Blackmond Laskey, K., "Model uncertainty: theory and practical implications," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 26, no. 3, pp. 340 – 348, 1996.

[24] Blum, C., "Ant colony optimization: Introduction and recent trends," *Physics of Life Reviews*, vol. 2, pp. 353–373, 2005.

[25] Borotschnig, H., Paletta, L., Prantl, M., and Pinz, A., "Comparison of probabilistic, possibilistic and evidence theoretic fusion schemes for active object recognition," *Computing*, vol. 62, no. 4, pp. 293 – 319, 1999.

[26] Buonanno, M. A., *A Method for Aircraft Concept Exploration Using Multicriteria Interactive Genetic Algorithms.* PhD thesis, Georgia Institute of Technology, 2005.

[27] Case, L., "No dummies here," *Automotive Industries*, vol. 186, no. 12, pp. 57–58, 2006.

[28] Chen, S.-Y., Lin, W.-C., and Chen, C.-T., "Evidential reasoning based on Dempster-Shafer theory and its application to medical image analysis," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 2032, pp. 35 – 46, 1993.

[29] Chen, X. and Lind, N. C., "Fast probability integration by three-parameter normal tail approximation," *Structural Safety*, vol. 1, no. 4, pp. 269 – 276, 1983.

[30] Christian, J. T. and Baecher, G. B., "Point-estimate method as numerical quadrature," *Journal of Geotechnical and Geoenvironmental Engineering*, vol. 125, no. 9, pp. 779 – 786, 1999.

[31] Clarke, S., Griebsch, J., and Simpson, T., "Analysis of support vector regression for approximation of complex engineering analysis," *Journal of Mechanical Design, Transactions of the ASME*, vol. 127, no. 6, pp. 1077–1087, 2005.

[32] Claus, R. W., Lavelle, T., Townsend, S., Turner, M., and Reed, J., "Variable fidelity analysis of complete engine systems." 43^{rd} AIAA / ASME / SAE /ASEE Joint Propulsion Conference and Exhibit, AIAA-2007-5042, July 2007.

[33] Coath, G. and Halgamuge, S., "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *The 2003 Congress on Evolutionary Computation*, vol. 4, pp. 2419–2425, 2003.

[34] Coello, C. A. C., "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, pp. 113–127, 2000.

[35] Coello, C. A. C., "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms - a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 1245–1287, 2002.

[36] Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., and Trubian, M., "Heuristics from nature for hard combinatorial optimization problems," *International Transactions in Operational Research*, vol. 3, no. 1, p. 1, 1996.

[37] Company, H. M., "The American Heritage Dictionary of the English Language." http://www.dictionary.com, January 2008.

[38] Converse, G. L., "Extended Parametric Representation of Compressor Fans and Turbines," Tech. Rep. NASA CR-174646, General Electric Company, 1984.

[39] Converse, G. L. and Griffin, R. G., "Extended Parametric Representation of Compressor Fans and Turbines," Tech. Rep. NASA CR-174645, General Electric Company, 1984.

[40] Converse, G., "Extended Parametric Representation of Compressor Fans and Turbines," Tech. Rep. NASA CR-174647, General Electric Company, 1984.

[41] Cormen, T., Leiserson, C., Rivest, R., and Stein, C., *Introduction to Algorithms*. McGraw-Hill Publishing Company, 2^{nd} ed., 2003.

[42] Cornell, C., "Probability-based structural code," *ACI Journal Proceedings*, vol. 66, no. 12, pp. 974 – 85, 1969.

[43] Crespo, L. G. and Kenny, S. P., "A first and second order moment approach to probabilistic control synthesis," in *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, vol. 4, pp. 2706 – 2739, 2005.

[44] Cumpsty, N., *Jet Propulsion - A Simple Guide to Aerodynamic and Thermodynamic Design and Performance of Jet Engines*. Cambridge University Press, 2^{nd} ed., 2003.

[45] Dantzig, G. B., "Discrete-variable extremum problems," *Operations Research*, vol. 5, no. 2, pp. 266–289, 1957.

[46] Dantzig, G. B. and Wolfe, P., "Decomposition principle for linear programs," *Operations Reserach*, vol. 8, no. 1, pp. 101–111, 1960.

[47] Dawson, S. and Kaufman, J., "The imperative for medical simulation," *Proceedings of the IEEE*, vol. 86, no. 3, pp. 479 – 83, 1998.

[48] Deb, K., "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 311–338, 2000.

[49] Deb, K., *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, Inc, 2001.

[50] Deb, K., Pratap, A., Agarwal, S., and Meyanivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evoluationary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[51] Dempster, A., "Upper and lower probabilities induced by a multivalued mapping," *Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, 1967.

[52] Deo, N. and Pang, C.-Y., "Shortest-path algorithms: Taxonomy and annotation," *Networks*, vol. 14, pp. 275–323, 1984.

[53] Devroye, L., *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[54] Dijkstra, E., "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[55] Ditlevsen, O., "Generalized second moment reliability index," *Journal of Structural Mechanics*, vol. 7, no. 4, pp. 435 – 451, 1979.

[56] Ditlevsen, O., "Principle of normal tail approxmation," *Journal of the Engineering Mechanics Division*, vol. 107, no. 6, pp. 1191 – 1208, 1981.

[57] Ditlevsen, O., "Model uncertainty in structural reliability," *Structural Safety*, vol. 1, pp. 73 – 86, 1982.

[58] Doerner, K., Gutjahr, W. J., Hartl, R. F., Strauss, C., and Stummer, C., "Ant colony optimization in multiobjective portfolio selection," in *Proceedings of the* 4^{th} *Metaheuristics International Conference*, 2001.

[59] Doerner, K., Gutjahr, W., Hart, R., Strauss, C., and Stummer, C., "Naure-inspired metaheuristics for multiobjective activity crashing," *Omega*, vol. 36, no. 6, pp. 1019–1037, 2008.

[60] Dorigo, M. and Stutzle, T., *Ant Colony Optimization*. 2002.

[61] Dorigo, M. and Caro, G. D., "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[62] Dorigo, M. and Caro, G. D., "Ant colony optimization: A new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 2, pp. 1470–1477, July 1999.

[63] Draper, D., "Assessment and propagation of model uncertainty," *Journal of the Royal Statistical Society, Series B*, vol. 57, no. 1, pp. 45–97, 1995.

[64] Du, X. and Chen, W., "Propagation and management of uncertainties in simulation-based collaborative systems design," in *Proceedings from The* 3^{rd} *World Congress of Structural and Multidisciplinary Design*, 1999.

[65] Dubois, D. and Prade, H., *Possibility Theory - An Approach to Computerized Processing of Uncertainty*. Plenum Press, 1988.

[66] E. M. Stearns, e. a., "Energy Efficient Engine Core Design and Performance Report," Tech. Rep. NASA CR-168069, General Electric Company, 1982.

[67] E. M. Stearns, e. a., "Energy Efficient Engine Integrated Core / Low Spool Design and Performance Report," Tech. Rep. NASA CR-168211, General Electric Company, 1985.

[68] Ehrgott, M. and Gandibleux, X., "A survey and annotated bibliogrpahy of multiobjective combinatorial optimization," *OR Spectrum*, vol. 22, no. 4, pp. 425–460, 2000.

[69] Ender, T. R., *A Top-Down, Hierarchical, System-of-Systems Approach to the Design of an Air Defense Weapon*. PhD thesis, Georgia Institute of Technology, 2006.

[70] Engelund, S. and Rackwitz, R., "Benchmark study on importance sampling techniques in structural reliability," *Structural Safety*, vol. 12, no. 4, pp. 255 – 276, 1993.

[71] Fisher, M. L., "Applications Oriented Guide to Lagrangian Relaxation," *Interfaces*, vol. 15, no. 2, pp. 10 – 21, 1985.

[72] Fisher, M., "The lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 27, no. 1, pp. 1 – 18, 1981.

[73] Fisher, M. and Shapiro, J., "Constructive duality in integer programming," *SIAM Journal of Applied Mathematics*, vol. 27, no. 1, pp. 31–52, 1974.

[74] Flores-Mendoza, J. I. and Mezura-Montes, E., "Dynamic adaptation and multiobjective concepts in a particle swarm optimizer for constrained optimization," in *The 2008 Congress on Evolutionary Computation*, pp. 3427–3434, 2008.

[75] Follen, G. and auBuchon, M., "Numerical Zooming Between a NPSS Engine System Simulation and a One-Dimensional High Compressor Analysis Code," Tech. Rep. NASA TM-2000-209913, NASA, 2000.

[76] Gabriel, S. A. and Bernstein, D., "The traffic equilibrium problem with nonadditive path costs," *Transportation Science*, vol. 31, no. 4, pp. 337–348, 1997.

[77] Gabriel, S. A. and Bernstein, D., "Nonadditive shortest paths: Subproblems in multiagent competitive network models," *Computational and Mathematical Organizaiton Theory*, vol. 6, no. 1, pp. 29–45, 2000.

[78] Gambardella, L. M., Taillard, R., and Agazzi, G., "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows," in *New Ideas in Optimization*, pp. 63–76, McGraw-Hill Publishing Company, 1999.

[79] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[80] Gomory, R., "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, no. 5, pp. 275–278, 1958.

[81] Gomory, R., "On the relation between integer and non-integer solutions to linear programs," *Proceedings of the National Academy of Science of the United States of America*, vol. 53, no. 2, pp. 260–265, 1965.

[82] Grandin, J.-F. and Moulin, C., "What practical differences between probabilities, possibilities and credibilities," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4731, pp. 86 – 97, 2002.

[83] Gross, J. and Yellen, J., *Graph Theory and its Applications*. CRC Press, 1999.

[84] Gu, X., Renaud, J., Batill, S., Brach, R., and Budhiraja, A., "Worst case propagated uncertainty of multidisciplinary systems in robust design optimization," *Structural and Multidisciplinary Optimization*, vol. 20, no. 3, pp. 190 – 213, 2000.

[85] Gu, X., *Uncertainty in Simulation-based Multidisciplinary Design Optimization*. PhD thesis, University of Notre Dame, 2003.

[86] Guntsch, M. and Middendorf, M., *Evolutionary Multi-Criteria Optimization*, vol. 2632 of *Lecture Notes in Computer Science*, ch. Solving Multi-Criteria Optimization Problems with Population-Based ACO, pp. 464–478. Springer Berlin and Heidelberg, 2003.

[87] Gupta, A. K. and Nadarajah, S., *Handbook of Best Distribution and its Applications*. CRC Press, 2004.

[88] Halton, J. H., "On the efficiency of certain quasi-random sequence of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2, pp. 84–90, 1960.

[89] Hammersley, J. M., "Monte-Carlo Methods for Solving Multivariable Problems," *Annals of the New York Academy of Sciences*, vol. 86, no. 3, pp. 844–874, 1960.

[90] Handler, G. and Zang, I., "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, no. 4, pp. 293 – 309, 1980.

[91] Harary, F., Norman, R. Z., and Cartwright, D., *Stuctural Models.* John Wiley and Sons, Inc, 1965.

[92] Harr, M. E., "Probabilistic estimates for multivariate analyses," *Applied Mathematical Modelling*, vol. 13, no. 5, pp. 313 – 318, 1989.

[93] Hart, P., Nilsson, N., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. SCC-4, no. 2, pp. 100 – 107, 1968.

[94] Hasofer, A. M. and Lind, N. C., "Exact and invariant second-moment code format," *Journal of the Engineering Mechanics Division*, vol. 100, no. EM1, pp. 111 – 121, 1974.

[95] Haykin, S., *Neural Networks: A Comprehensive Foundation.* Prentice Hall, Inc, 2^{nd} ed., 1999.

[96] Hayter, A. J., *Probability and Statistics for Engineers and Scientists.* Duxbury, 2^{nd} ed., 2002.

[97] He, Q., Wang, L., and Huang, F.-Z., "Nonlinear constrained optimization by enhanced co-evolutionary pso," in *Conference on Evolutionary Computation*, vol. 1, pp. 83–89, 2008.

[98] Heidelberger, P., "Fast simulation of rare events in queueing and reliability models," *ACM Transactions on Modeling and Computer Simulation*, vol. 5, no. 1, pp. 43 – 85, 1995.

[99] Held, M., Wolfe, P., and Crowder, H., "Validation of subgradient optimization," *Mathematical Programming*, vol. 6, no. 1, pp. 62 – 88, 1974.

[100] Helton, J., "Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal," *Reliability Engineering and System Safety*, vol. 42, no. 2-3, pp. 327 – 67, 1993.

[101] Helton, J. and Davis, F., "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Reliability Engineering and System Safety*, vol. 81, no. 1, pp. 23 – 69, 2003.

[102] Helton, J., Johnson, J., Oberkampf, W., and Storlie, C., "A sampling-based computational strategy for the representation of epistemic uncertainty in model predictions with evidence theory," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 37-40 SPEC ISS, pp. 3980 – 3998, 2007.

[103] Helton, J. and Oberkampf, W., "Alternative representations of epistemic uncertainty," *Reliability Engineering and System Safety*, vol. 85, no. 1-3, pp. 1 – 10, 2004.

[104] Hill, P. and Peterson, C., *Mechanics and Thermodynamics of Propulsion.* Addison-Wesley Publishing Company, 2^{nd} ed., 1992.

[105] Hillier, F. S. and Lieberman, G. J., *Introduction to Operations Research*. McGraw-Hill Publishing Company, 5^{th} ed., 1990.

[106] Hoffman, J. C. and Murphy, R. R., "Comparison of bayesian and dempster-shafer theory for sensing: a practitioner's approach," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 2032, pp. 266 – 279, 1993.

[107] Holloway, P. R., Knight, G. L., Koch, C. C., and Shaffer, S. J., "Energy Efficient Engine High Pressure Compressor Detail Design Report," Tech. Rep. NASA CR-165558, General Electric Company, 1982.

[108] Hong, H., "Efficient point estimate method for probabilistic analysis," *Reliability Engineering and System Safety*, vol. 59, no. 3, pp. 261 – 267, 1998.

[109] Howard, R., "Uncertainty about probability - a decision-analysis perspective," *Risk Analysis*, vol. 8, no. 1, pp. 91–98, 1988.

[110] Hu, X. and Eberhart, R., "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Congress on Evolutionary Computation*, vol. 2, pp. 1677–1681, 2002.

[111] Ibrahim, Y., "Observations on applications of importance sampling in structural reliability analysis," *Structural Safety*, vol. 9, no. 4, pp. 269 – 281, 1991.

[112] Iman, R. and Conover, W., "Small sample sensitivity analysis techniques for computer models, with an application to risk assessment," *Communications in Statistics - Theory Methods*, vol. A9, no. 17, pp. 1749 – 842, 1980.

[113] John, J. E. A., *Gas Dynamics*. Prentice Hall, Inc, 2^{nd} ed., 1984.

[114] Joksch, H., "The shortest route problem with constraints," *Journal of Mathematical Analysis and Applications*, vol. 14, no. 2, pp. 191 – 197, 1966.

[115] Karaboga, D. and Basturk, B., "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[116] Kass, R. E. and Raftery, A. E., "Bayes factors," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 773–795, 1995.

[117] Keeman, V., *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. MIT Press, 2001.

[118] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm Intellgence*. Morgan Kaufmann Publishers, 2001.

[119] King, R. T. F. A. and Rughooputh, H. C. S., "Elitist multiobjective evolutionary algoirthm for environmental/economic dispatch," in *The 2003 Congress on Evolutionary Computation*, vol. 2, pp. 1108–1114, 2003.

[120] Kirby, M., Barros, P., and Mavris, D., "Enhancing the environmental policy making process with the faa's eds analysis tool." 47^{th} AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2009-1262, January 2009.

[121] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[122] Klir, G., "On the alleged superiority of probabilistic representation of uncertainty," *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 1, pp. 27 – 31, 1994.

[123] Klir, G. J. and Wierman, M. J., *Uncertainty-Based Information - Elements of Generalized Information Theory*. Physica-Verlag, 2^{nd} ed., 1999.

[124] Kocis, L. and Whiten, W. J., "Computational investigations of low-discrepancy sequences," *ACM Transactions on Mathematical Software*, vol. 23, no. 2, pp. 266 – 294, 1997.

[125] Korkmaz, T. and Krunz, M., "Multi-constrained optimal path selection," in *Proceedings IEEE INFOCOM 2001 Conference on Computer Communications Twentieth Annual Joint Conference of the IEEE Computer and Communications Society*, vol. 2, pp. 834–843, 2001.

[126] Kramer, O., Brugger, S., and Lazovic, D., "Sex and death: Towards biologically inspired heuristics for constraint handling," in *Proceedings from the 9th Annual Conference on Genetic and Evolutionary Computation*, pp. 666–673, 2007.

[127] Kurdi, M. H., Haftka, R. T., Schmitz, T. L., and Mann, B. P., "A numerical study of uncertainty in stability and surface location in high-speed milling," in *ASME In- Proceedings from the Iternational Mechanical Engineering Congress and Exposition*, vol. 16-1, pp. 387 – 395, 2005.

[128] Land, A. and Doig, A., "An automatic method for solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[129] Lawler, E., "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Management Science*, vol. 18, no. 7, pp. 401 – 5, 1972.

[130] Lawler, E. L., *Combinatorial Optimization - Networks and Matroids*. Hold, Rinehart and Winston, 1976.

[131] Lee, I.-H., Shin, S.-Y., and Zhang, B.-T., "Dna sequence optimization using constrained multi-objective evolutionary algorithm," in *The 2003 Congress on Evolutionary Computation*, vol. 4, pp. 2270–2276, 2003.

[132] Lewis, P. A. W. and Orav, E. J., *Simulation Methodology for Statisticians, Operations Analysts, and Engineers*, vol. 1. Wadsworth and Brooks, 1989.

[133] Li, K., "Point-estimate method for calculating statistical moments," *Journal of Engineering Mechanics*, vol. 118, no. 7, pp. 1506 – 1511, 1992.

[134] Liang, Y.-C. and Smith, A. E., "An ant colony optimization algorithm for the redundancy allocation problem," *IEEE Transactions on Reliability*, vol. 53, pp. 417–422, Spetember 2004.

[135] Light, W., *Ridge Functions, Sigmoidal Functions, and Neural Networks*. Academic Press, 1990.

[136] Lind, N. C., "Modelling of uncertainty in discrete dynamical systems.," *Applied Mathematical Modelling*, vol. 7, no. 3, pp. 146 – 152, 1983.

[137] Littlewood, D. J., Drakopoulos, P. A., and Subbarayan, G., "Pareto-optimal formulations for cost versus colorimetric accuracy trade-offs in printer color management," *ACM Transactions on Graphics*, vol. 21, no. 2, 2002.

[138] Lytle, J. K., "The Numerical Propulsion System Simulation - A Multidisciplinary Design System for Aerospace Vehicles," Tech. Rep. NASA TM-1999-209194, NASA-Glenn, 1999.

[139] Madigan, D. and Raftery, A. E., "Model selection and accounting for model uncertainty in graphical models using ocean's window," *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1535–1546, 1994.

[140] Magnanti, T., Shapiro, J., and Wagner, M., "Generalized linear programming solves the dual," *Management Science*, vol. 22, no. 11, pp. 1195–1203, 1976.

[141] Mahfouf, M., Chen, M.-Y., and Linkens, D. A., *Parallel Problem Sovling from nature*, vol. 3242 of *Lecture Notes in Computer Science*, ch. Adaptive weighted particle swarm optimization for multi-objective optimal design of alloy steels, pp. 762–771. Springer-Verlag, 2004.

[142] Mantis, G. C., *Quantification and Propagation of Disciplinary Uncertainty via Bayesian Statistics*. PhD thesis, Georgia Institute of Technology, 2002.

[143] Marsaw, A., German, B., Hollingsworth, P., Mavris, D., and Csonka, S., "Adaptive selection of engine technology solutions sets from a large combinatorial space." 45^{th} AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2007-1333, 2007.

[144] Mattingly, J. D., *Elements of Gas Turbine Propulsion*. AIAA, 2005.

[145] Mavris, D., Bandte, O., and Delaurentis, D., "Robust design simulation: A probabilitistic approach to multidisciplinary design," *AIAA Journal of Aircraft*, vol. 36, no. 1, p. 298, 1999.

[146] McDonald, R., *Error Propagation and Metamodeling for a Fidelity Tradeoff Capability in Complex Systems Design*. PhD thesis, Georgia Institute of Technology, 2006.

[147] McKenna, T. M., *The Role of Interdisciplinary Research Involving Neuroscience in the Development of Intelligent Systems*. Academic Press, 1994.

[148] Mehlhorn, K. and Ziegelmann, M., *Lecture Notes in Computer Science*, ch. Resource Constrained Shortest Paths. Springer Berlin and Heidelberg, 2000.

[149] Melchers, R., "Importance sampling in structural systems," *Structural Safety*, vol. 6, no. 1, pp. 3 – 10, 1989.

[150] Melchers, R., "Improved importance sampling methods for structural system reliability calculation," in *Proceedings of ICOSSAR 1989, the* 5^{th} *International Conference on Structural Safety and Reliability*, pp. 1185 – 1192, 1989.

[151] Merkle, D., Middendorf, M., and Schmeck, H., "Ant colony optimization for resource-constrained project scheduling," *IEEE Transactions on Evoluationary Computation*, vol. 6, no. 4, pp. 333–346, 2002.

[152] Metropolis, N. and Ulam, S., "The monte carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[153] Michalewicz, Z., "A survey of constraint handling techniques in evolutionary computation methods," in *Proceedings from the 4th Annual Conference on Evolutionary Programming*, pp. 135–155, MIT Press, 1995.

[154] Michelon, P. and Maculan, N., "Lagrangean decomposition for integer nonlinear programmign with linear constraints," *Mathematical Programming*, vol. 52, pp. 303–313, 1991.

[155] Miller, A.C., I. and Rice, T., "Discrete approximations of probability distributions," *Management Science*, vol. 29, no. 3, pp. 352 – 62, 1983.

[156] Millwater, H., Wu, Y.-T., Dias, J., McClung, R., Raveendra, S., and Thacker, B., "Nessus software system for probabilistic structural analysis," in *Proceedings of ICOSSAR 1989, the* 5^{th} *International Conference on Structural Safety and Reliability*, pp. 2283 – 2290, 1989.

[157] Morales, A. K. and Quezada, C. V., "A universal eclectic gnetic algorithm for constrained optimization," in *Proceedings from the 3rd International Conference on Natural Computation 6th European Congress on Intelligent Techniques and Soft Computing*, pp. 518–522, 1998.

[158] Muller, B. and Reinhardt, J., *Neural Networks: An Introduction*. Springer-Verlag, 1990.

[159] Myers, R. H. and Montgomery, D. C., *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley and Sons, Inc, 2002.

[160] Nakrani, S. and Tovey, C., "On honey bees and dynamic server allocation in internet hosting centers," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 223–240, 2004.

[161] National Institute of Standards and Technology and SEMATECH, "Engineering Statistics Handbook." http://www.itl.nist.gov/div898/handbook/pri/section7/pri7.htm, January 2008.

[162] National Science Foundation Blue Ribbon Panel, *Simulation-Based Engineering Science: Revolutionizing Engineering Science through Simulation*, 2006.

[163] Niederreiter, H., *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992.

[164] Niederreiter, H., "Quasi-monte carlo methods and pseudo random numbers," *Bulletin of the American Mathematical Society*, vol. 84, no. 6, pp. 957–1041, 1978.

[165] Nikolaidis, E., Chen, S., Cudney, H., Haftka, R. T., and Rosca, R., "Comparison of probability and possibility for design against catastrophic failure under uncertainty," *Journal of Mechanical Design, Transactions of the ASME*, vol. 126, no. 3, pp. 386 – 394, 2004.

[166] Nixon, J. N., *A Systematic Process for Adaptive Concept Exploration*. PhD thesis, Georgia Institute of Technology, 2006.

[167] Oberkampf, W. L., DeLand, S. M., Rutherford, B. M., Diegert, K. V., and Alvin, K. F., "New methodology for the estimation of total uncertainty in computational simulation," in *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, vol. 4, pp. 3061 – 3083, 1999.

[168] Oberkampf, W. L., DeLand, S. M., Rutherford, B. M., Diegert, K. V., and Alvin, K. F., "Error and uncertainty in modeling and simulation," *Reliability Engineering and System Safety*, vol. 75, no. 3, pp. 333 – 357, 2002.

[169] Oberkampf, W. L., Helton, J. C., Joslyn, C. A., Wojtkiewicz, S. F., and Ferson, S., "Challenge problems: Uncertainty in system response given uncertain parameters," *Reliability Engineering and System Safety*, vol. 85, no. 1-3, pp. 11 – 19, 2004.

[170] Park, J.-B., Lee, K.-S., Shin, J.-R., and Lee, K. Y., "A particle swarm optimization for economic dispatch with nonsmooth cost functions," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 34–42, 2005.

[171] Parker, R. G. and Rardin, R. L., *Discrete Optimization*. Academic Press, 1988.

[172] Parsopoulos, K. E., Tasoulis, D. K., and Vrahatis, M. N., "Multiobjective optimization using parallel, vector evaluated particle swarm optimization," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, vol. 2, pp. 823–828, 2004.

[173] Parsopoulos, K. E. and Vrahatis, M. N., "Particle swarm optimization method in multiobjective problems," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, pp. 603–607, 2002.

[174] Pearson, E., "William Sealy Gosset: 'Student' as a Statistician," *Biometrika*, vol. 30, pp. 210–250, 1939.

[175] Plane, D. R. and McMillan, C., *Discrete Optimization: Integer Programming and Network Analysis for Management Decisions*. Prentice Hall, Inc, 1971.

[176] Pourtakdoust, S. H. and Nobahari, H., *Ant Colony, Optimization and Swarm Intelligence*, vol. 3172 of *Lecture Notes in Computer Science*, ch. An Extension of Ant Colony System to Continuous Optimization Problems, pp. 294–301. Springer Berlin and Heidelberg, 2004.

[177] Prior, M., Stults, I., Daskilewicz, M., Duncan, S., German, B., and Mavris, D., "A framework for methods incorporating high-fidelity physics-based models into the conceptual design of combined cycle power plants." ASME 3^{rd} Conference on Energy Sustainability, ES2009-90291, 2009.

[178] Rallabhandi, S. K. and Mavris, D., "Simultaneous airframe and propulsion cycle optimization for supersonic aircraft design," *AIAA Journal of Aircraft*, vol. 45, no. 1, pp. 38–55, 2008.

[179] Reyes-Sierra, M. and Coello, C. A. C., "Improving pso-based multi-objective optimization using crowding, mutation, and ϵ dominance," in *Third International Conference on Evolutionary Multi-Criterion Optimization*, pp. 505–519, Springer-Verlag, 2005.

[180] Reyes-Sierra, M. and Coello, C. A. C., "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[181] Ripley, B. D., *Stochastic Simulation*. John Wiley and Sons, Inc, 1987.

[182] Rosenblatt, M., "Remarks on multivariate transformation," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 470–472, 1952.

[183] Rosenblueth, E., "Two-point estimates in probabilities," *Applied Mathematical Modelling*, vol. 5, no. 5, pp. 329 – 35, 1981.

[184] Rosenblueth, E., "Point estimates for probability moments," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 72, pp. 3812–3814, 1975.

[185] Roth, B. A., Doel, D. L., and Cissell, J. J., "Probabilistic matching of turbofan engine performance models to test data." ASME Turbo Expo 2005: Land Sea and Air, GT2005-68201, 2005.

[186] Saigal, R., "A constrained shortest route problem," *Operations Research*, vol. 16, no. 205-209, 1968.

[187] Sampath, R., Irani, R., Balasubramaniam, M., and Plybon, R., "High fidelity system simulation of aerospace vehicles using npss." 42^{nd} AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2004-371, January 2004.

[188] Schoenauer, M. and Xanthakis, S., "Constrained ga optimization," in *Proceedings from the 5th International Conference on Genetic Algorithms*, pp. 573–580, 1993.

[189] Scott, W. B., "Virtual thrust," *Aviation Week and Space Technology*, vol. 166, no. 2, p. 112, 2007.

[190] Shafer, G., *A Mathematical Theory of Evidence*. Princeton, 1976.

[191] Shaffer, D., Meglan, D., Ferrell, M., and Dawson, S., "Virtual rounds: Simulation-based education in procedural medicine," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 3712, pp. 99 – 108, 1999.

[192] Shelokar, P. S., Jayaraman, V., and Kulkarni, B., "Ant algorithm for single and multiobjective reliability optimization problems," *Quality and Reliability Engineering International*, vol. 18, no. 6, pp. 497 – 514, 2002.

[193] Shier, D. R., "On algorithms for finding the k shortest paths in a network.," *Networks*, vol. 9, no. 3, pp. 195 – 214, 1979.

[194] Sivakumar, R. A. and Batta, R., "The variance-constrained shortest path problem," *Transportation Science*, vol. 28, no. 4, pp. 309–316, 1994.

[195] Smets, P. and Kennes, R., "The transferable belief model," *Artificial Intelligene*, vol. 66, no. 2, pp. 191 – 234, 1994.

[196] Smith, P. J., Shafi, M., and Gao, H., "Quick simulation: a review of importance sampling techniques in communications systems," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 4, pp. 597 – 613, 1997.

[197] Sobieszczanski-Sobieski, J., "Sensitivity of complex, internally coupled systems," *AIAA Journal*, vol. 28, no. 1, pp. 153 – 160, 1990.

[198] Socha, K., *Ant Colony, Optimization and Swarm Intelligence*, vol. 3174 of *Lecture Notes in Computer Science*, ch. ACO for Continuous and Mixed-Variable Optimization, pp. 25–36. 2004.

[199] Spero, E., Blodgett, K., Allen, L., Anzalone, E., and Hendricks, E., "The conceptual design of a hybrid target system for missile defense testing needs." AIAA Graduate Team Missile Design Competition Final Report, 2007.

[200] Spindler, P. M., "Environmental design space model assessment," Master's thesis, Purdue University, 2005.

[201] Stauber, L. J. and Naiman, C. G., "Numerical Propuslion System Simulation (NPSS): An Award Winning Propulsion System Simulation Tool," NASA Technical Report 20050214739, NASA-Glenn, 2001.

[202] Stentz, A., "Optimal and efficient path planning for partially-known environments," in *Proceedings 1994 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310–3317, IEEE, 1994.

[203] Stults, I., Wilson, J. S., Beisecker, E., Morris, S., Leahu-Aluas, I., and Mavris, D., "Conceptual Design of CELESTE: a Cost-Effiective Low-Noise, SBJ Turbofan Engine." 41^{st} AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, AIAA-2005-3845, July 2005.

[204] Sun, Z.-G. and Teng, H.-F., "Optimal layout design of a satellite module," *Engineering Optimization*, vol. 35, no. 5, pp. 513–529, 2003.

[205] Suppes, P., "The measurement of belief," *Journal of the Royal Statistical Society, Series B*, vol. 36, no. 2, pp. 160–191, 1974.

[206] (TASS), T. A. S. S., "Tass: Delivering transport safety solutions." http://www.tass-safe.com.

[207] Teichroew, D., "A history of distribution sampling prior to the era of the computer and its relevance to simulation," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 27–49, 1965.

[208] Timko, L. P., "Energy Efficient Engine High Presure Turbine Component Test Performance Report," Tech. Rep. NASA CR-168289, General Electric Company, 1982.

[209] Tsaggouris, G. and Zaroliagis, C., *Algorithms*, ch. Non-Additive Shortest Paths. Springer Berlin and Heidelberg, 2004.

[210] US National Research Council Committee on Aeronautical Technologies, *Aeronautical Technologies for the Twenty-First Century*. National Academies Press, 1992.

[211] Uyttenhove, K. and Steyaert, M. S., "Speed-power-accuracy tradeoff in high-speed cmos adcs," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 4, pp. 280 – 287, 2002.

[212] Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design*. Vanderplaats Research and Development, Inc, 3^{rd} ed., 2001.

[213] Vapnik, V., *Statistical Learning Theory: Adaptive and Learning Systems for Signal Processing, Communications, and Control*. John Wiley and Sons, Inc, 1998.

[214] Villeneuve, F., *A Method for Concept and Technology Exploration of Aerospace Architectures*. PhD thesis, Georgia Institute of Technology, 2007.

[215] Wallace, J., Volovoi, V., and Mavris, D., "A method for modeling system-driven uncertainty during probabilistic part life analyses." 40^{th} AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, AIAA-2004-3989, July 2004.

[216] Wang, L., Beeson, D., and Wiggs, G., "A robust and efficient probabilistic approach for challenging industrial applications with high-dimensional and non-monotonic design spaces," in *Collection of Technical Papers - 11^{th} AIAA/ISSMO Multidisciplinary Analysis and Optimization Conferenc*, vol. 2, pp. 1211 – 1224, 2006.

[217] Wang, L., Beeson, D., and Wiggs, G., "Efficient moment and probability distribution estimation using the point estimate method for high-dimensional engineering problems," in *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, vol. 2, pp. 2093 – 2110, 2007.

[218] Wellman, M. P., *Uncertainty in Artificial Intelligence*, ch. Qualitative Probabilistic Networks for Planning Under Uncertainty. Elsevier Science, 1987.

[219] Whittle, P., *Optimization Under Constraints: Theory and Applications of Nonlinear Programming*. Wiley-Interscience, 1971.

[220] Wong, F. S., "First-order, second-moment methods," *Computers and Structures*, vol. 20, no. 4, pp. 779 – 791, 1985.

[221] Wu, C. F. J. and Hamada, M., *Experiments: Planning, Analysis, and Parameter Design Optimization*. John Wiley and Sons, Inc, 2000.

[222] Wu, J., Apostolakis, G., and Okrent, D., "Uncertainties in system analysis. probabilistic versus nonprobabilistic theories," *Reliability Engineering and System Safety*, vol. 30, no. 1-3, pp. 163 – 181, 1990.

[223] Wu, Y.-T., *Efficient Methods for Mechanical and Structural Reliability Analysis and Design*. PhD thesis, University of Arizona, 1984.

[224] Wu, Y.-T., "Demonstration of a new, fast probability integration methods for reliability analysis," *Journal of Engineering for Industry, Transactions of the ASME*, vol. 109, no. 1, pp. 24 – 28, 1987.

[225] Xu, Y., Liu, C., and Li, D., "Generalized nonlinear lagrangian formulation for bounded integer programming," *Journal of Global Optimization*, vol. 33, no. 2, pp. 257–272, 2005.

[226] Yang, X.-S., *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, ch. Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. Lecture Notes in Computer Science, Springer Berlin and Heidelberg, 2005.

[227] Yang, X.-S., *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[228] Yang, X.-S., *Music-Inspired Harmony Search Algorithm*, vol. 191, ch. Harmony Search as a Metaheuristic Algorithm, pp. 1–14. Springer Berlin and Heidelberg, 2009.

[229] Yen, J., "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712 – 16, 1971.

[230] Zadeh, L., "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Systems*, vol. 1, no. 1, pp. 3 – 28, 1978.

[231] Zhao, Y.-G. and Ono, T., "New point estimates for probability moments," *Journal of Engineering Mechanics*, vol. 126, no. 4, pp. 433 – 436, 2000.

[232] Zheng, J., Wu, Q., and Song, W., "An improved particle swarm algorithm for solving nonlinear constrained optimization problems," in *Proceedings from the 3rd International Conference on Natural Computation*, vol. 4, pp. 112–117, 2007.

[233] Zoppou, C. and Li, K., "New point estimate method for water resources modeling," *Journal of Hydraulic Engineering*, vol. 119, no. 11, pp. 1300 – 1307, 1993.